Results from the project AURES: Autonomous UGV-system for Reconnaissance and Surveillance

P. ÖGREN (EDITOR), D. ANISI, D. BERGLUND, D. DIMAROGONAS, H. GUSTAVSSON, L. HEDLIN, J. HEDSTRÖM, X. HU, K. H. JOHANSSON, F. KATSILIERIS, V. KAZNOV, P. LIF, M. LINDHÉ, U. NILSSON, M. PERSSON, M. SEEMAN, P. SVENMARCK, J. THUNBERG

FOI-R--2783--SE User report
ISSN 1650-1942 April 2009

Defence and Security, Systems and Technology

P. Ögren (Editor), D. Anisi, D. Berglund, D. Dimarogonas,
H. Gustavsson, L. Hedlin, J. Hedström, X. Hu, K. H. Johansson,
F. Katsilieris, V. Kaznov, P. Lif, M. Lindhé, U. Nilsson, M. Persson,
M. Seeman, P. Svenmarck, J. Thunberg

Results from the project AURES: Autonomous UGV-system for Reconnaissance and Surveillance

Titel	Resultat från projektet AURES: Autonoma UGV-system för Spaning och Bevakning
Title	Results from the project AURES: Autonomous UGV-system for Reconnaissance and Surveillance
Rapportnummer / Report no	FOI-R--2783--SE
Rapporttyp / Report type	Användarrapport / User report
Utgivningsår / Year	2009
Antal sidor / Pages	100
Kund / Customer	FMV, 297316-LB704859
Forskningsområde	5. Bekämpning och skydd
Research area	5. Strike and Protection
Delområde	51. Vapen och skydd
Sub area code	51. Weapons and Protection
Projektnummer / Project no	E28004
Godkänd av / Approved by	Lars Höstbäck Head, Defence and Security, Systems and Technology
ISSN	ISSN-1650-1942

FOI Swedish Defence Research Agency
Defence and Security, Systems and Technology
SE-164 90 STOCKHOLM

Abstract

In this document, the results from the project AURES are reported. The aim of AURES is to develop, demonstrate and evaluate high level autonomous capabilities in a UGV system for reconnaissance and surveillance. The core functionality of a UGV surveillance system is to supply its operators with information, resulting in good *situational awareness*, i.e., knowledge about what goes on in the area of interest. To supply this information, most UGV systems are equipped with cameras, and the quality of the resulting awareness depends to a large extent on where those cameras are, when they are there, and what they are pointed at. This is the very focus area of AURES: how to move and point the UGV cameras over time to enhance the situational awareness of the operators.
In detail, the results in this report include the following. Algorithms for positioning of the UGVs such that the cameras provide a concurrent *line-of-sight perimeter* surrounding a given object of interest. Algorithms for positioning of the UGVs such that the cameras provide concurrent video *coverage of all designated walls*, or objects of interest. A new patented *teleoperation control mode* that allows the operator to focus more on where he wants the UGV camera to be, than how the UGV should move in order to get it there. Algorithms for UGV motions that *extend the effective range* of a wireless network. Algorithms for *efficiently patrolling* an area in short time and finally algorithms for *searching and securing* an area to make sure no evader can remain unseen. Included in the document are also descriptions of the hardware, including the Groundbot, and software used, as well as the results of user discussions and evaluations carried out within the project. This report aims at being accessible to a wide range of readers, not only robotic researchers, but also potential UGV system users and other UGV stake holders.

AURES started in January 2007 and finished in April 2009. The authors have the following affiliations. FOI: P. Ögren, J. Hedström, P. Lif, U. Nilsson, P. Svenmarck, KTH: D. Anisi, D. Dimarogonas, X. Hu, K. H. Johansson, F. Katsilieris, M. Lindhé, J. Thunberg Saab: D. Berglund, H. Gustavsson, L. Hedlin, M. Persson, Rotundus: V. Kaznov, M. Seeman,

Keywords

övervakning, spaning, UGV, robotik, autonomi, samverkan, banplanering, uppgiftstilldelning

Sammanfattning

Denna rapport beskriver resultaten från projektet AURES. Målet med AURES är att utveckla, demonstrera och utvärdera högnivå-förmågor i ett UGV-system för bevakning och spaning. Kärnfunktionen i ett UGV-bevakningssystem är att förse dess operatörer med en god bild av vad som händer i ett område. För att skapa denna lägesbild är de flesta UGV-system försedda med kameror, och kvaliten på lägesbilden beror i stor utsträckning på var kamerorna är, när de är där, och åt vilket håll de är riktade. Detta är AURES fokusområden: hur skall kamerorna röras och riktas över tiden för att möjliggöra en god lägesbild. I detalj innehåller rapporten följande. Algoritmer för att skapa en line-of-sight inringning av ett område av intresse. Algoritmer för att skapa samtidig kameratäckning av ett antal väggar och byggnader av intresse. En ny patenterad styrmod för fjärrstyrning (teleoperation) av UGV:er som låter operatören fokusera på hur han vill att kameran skall röra sig, snarare än hur roboten skall åka för att åstadkomma detta. Algoritmer för att utöka den effektiva räckvidden för trådlös överföring av videoströmmar från UGV:n. Algoritmer för att effektivt patrullera ett område på kort tid och slutligen algoritmer för att söka av och säkra ett område för att säkerställa att ingen oupptäckt individ befinner sig där. Texten innehåller också beskrivning av hårdvara, inklusive Groundbot, och mjukvara som använts samt resultat från användardiskussioner och utvärderingar. Rapporten är tänkt att vara läsbar inte bara för robotforskare, utan också för framtida UGV användare och andra intressenter.

Nyckelord

övervakning, spaning, UGV, robotik, autonomi, samverkan, banplanering, uppgiftstilldelning

Contents

1 Introduction

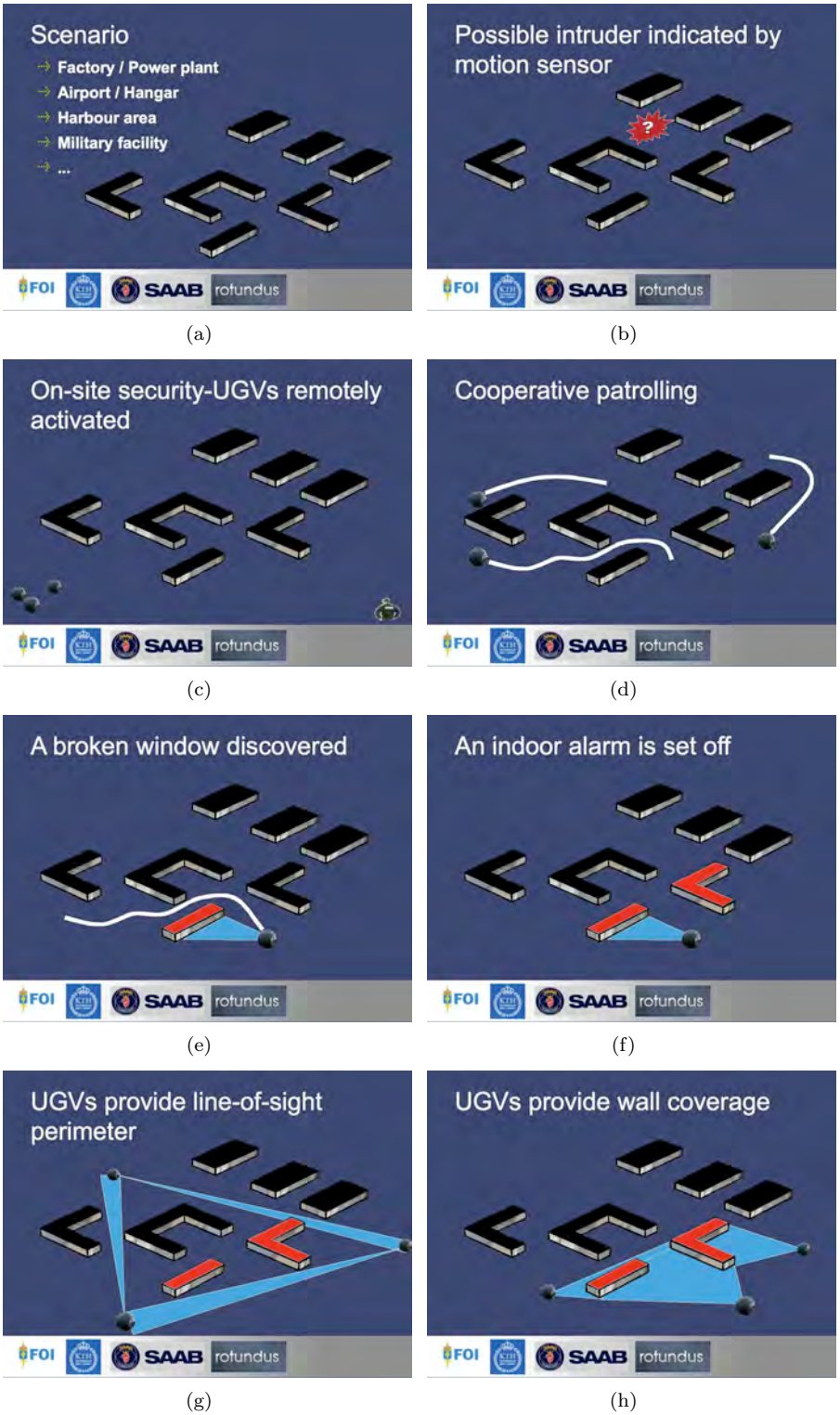
AURES is a joint project between FOI, KTH, Saab Aerotech and Rotundus. The aim of AURES is to provide high level autonomous capabilities to unmanned ground vehicle (UGV) systems performing surveillance and reconnaissance. An example of such a system is the Rotundus Groundbot, seen in Figure 1.1. We begin this chapter with describing a scenario involving such a UGV system and then discuss some variations of the scenario and its relevance from a user point of view. We then state what research areas are within the scope of the project and what areas are not. Finally, we list the scientific contributions and publications produced, and provide an outline of the rest of the report.

Figure 1.1: The Groundbot, developed by Rotundus AB.

1.1 Scenario

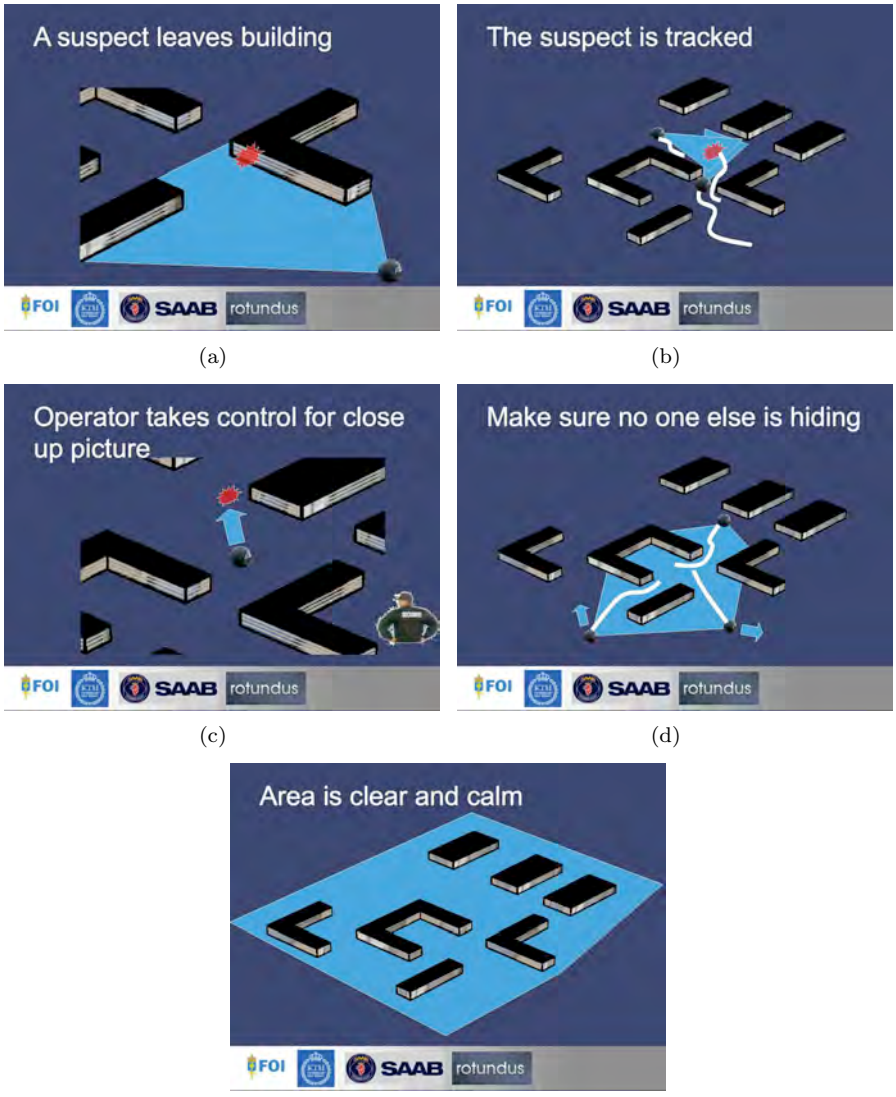
The surveillance scenario we have been using is illustrated in the 13 snapshots of Figures 1.2 and 1.3. The setting shown in Figure 1.2(a) corresponds to either a factory, a power plant, some hangars at an airport, a harbor area or a military facility of some sort, e.g. a military base, a decentralized equipment storage, or an international camp. At the beginning of the scenario, in Figure 1.2(b), a motion sensor indicates that some unusual activity is taking place at the facility. The on-site surveillance UGVs are then remotely activated by an operator situated at the regional control center of the security company, 1.2(c).

The operator has a graphical user interface (GUI) similar to the one depicted in Figure 1.4(a-b) to control all UGVs and any available stationary surveillance cameras. To increase the level of security in response to the motion sensor indication the operator designates a number of UGVs, under *Resources*, to be controlled as a group (Figure 1.4(b), right) and then picks *Patrolling* under *Group Tasks* and clicks *Start Algorithm*. The designated UGVs then proceed to patrol the area in a way that covers the whole area in minimum time, Figure 1.2(d). During the patrolling, a broken window is seen by one of the UGVs, 1.2(e), and then an indoor alarm is set of in another building, 1.2(f). The operator decides to monitor the two buildings in detail. He first



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 1.2: The first eight illustrations of the scenario.

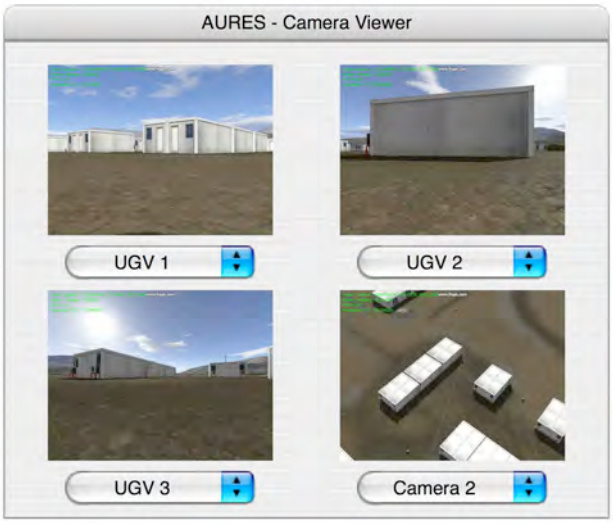


(a) (b) (c) (d) (e)

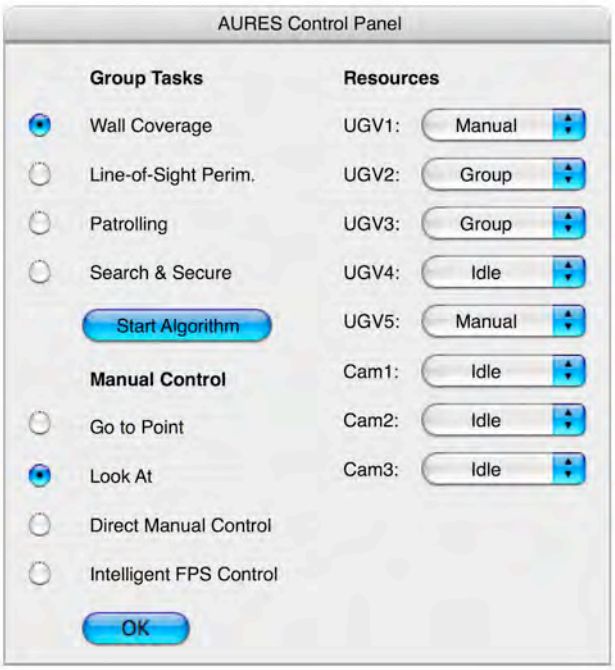
Figure 1.3: The last five illustrations of the scenario.

picks *Line-of-Sight Perimeter* in the GUI, and a minimal number of UGVs move to create a virtual barrier in terms of a line-of-sight perimeter around the two buildings of interest, Figure 1.2(g). In order to create the perimeter with only three UGVs a third building is included inside the perimeter. While keeping the perimeter, the operator decides to check if the available number of UGVs, i.e., three, is enough to create a concurrent coverage of all walls of the two buildings. The algorithms find such a solution and the operator decides to apply it, Figure 1.2(h).

While the two buildings are under close surveillance, the operator can now let any additional UGVs continue patrolling, or perform some other task. After a while, one of the video streams show an intruder exiting one of the buildings through a window, Figure 1.3(a). The operator decides to let the group of UGVs autonomously track the suspect to keep him in view, Figure 1.3(b), and



(a)



(b)

Figure 1.4: A possible UGV control software GUI. Note that *Intelligent FPS Control* is the same thing as *Enhanced Teleoperation* above, see chapter 4.

then takes manual control of one of the UGVs to get a close up picture of the perpetrator, Figure 1.3(c), or perhaps talk to him through the UGV speakers. After making sure that the suspect is either safely escorted off the premises, or tracked until manned security arrives, the operator gives a *Search and Secure* command to the UGV group to make sure no one else is hiding inside the facility, Figure 1.3(d,e). Note that this last option differs from patrolling in

that it guarantees that even a moving suspect can not remain unseen in the area. This higher ambition is reflected by the fact that securing an area often requires more UGVs than just patrolling it.

1.1.1 Variations of the Scenario

The above scenario is just en example of a situation in which a UGV surveillance system might prove valuable. Military reconnaissance is another area where many relevant scenarios can be created, and the enhanced teleoperation result in chapter 4 can naturally be used in any search-and-rescue or bomb-disposal mission. As for the scenario above, we would like to list the following variations. A changing environment, such as a harbor area, makes flexible UGVs even more attractive compared to stationary sensors. We can thus easily imagine some of the buildings in Figures 1.2 and 1.3 replaced by temporarily stored valuable containers. Furthermore, in Figure 1.2(b) the indication might be either a motion sensor, any other type of sensor, or a human calling security to report suspicious activity. In Figure 1.2(c) the UGVs might be on-site, or they could be brought there in the trunk of a car by a security guard using a mobile control pad to operate the UGVs. In Figure 1.2(e) the video streams of the UGVs might either be watched by operators at the control center, or pre-screened by computer vision algorithms, calling the operators attention to parts of the scene that differ from the normal in some sense. In Figure 1.3(c) the operator might take a close up picture, or engage the suspect with some kind of non-lethal weapon, such as tear-gas or spray paint, to enable later capture by human security guards.

1.1.2 Relevance of Scenario

As part of the AURES project we have conducted user interviews with security experts from the company Securitas, and different parts of the Swedish Armed Forces, e.g., F17, P4, K3, FHS (Försvarshögskolan) and MSS (Markstridsskolan). The overall results of those interviews where that UGVs are an interesting option for security and surveillance tasks. In both civil and military applications, there is a general trend towards fewer but more qualified and better equipped personnel. In many cases stationary surveillance equipment such as cameras and motion sensors can be used to solve the problem. However, in cases where valuable items are temporarily stored somewhere, such as an airport or harbor area, or before a major military exercise or mission, the UGV solution is promising. Furthermore, the military security experts saw a potential use for UGV both in surveillance of known facilities, and as support for a unit entering a new area e.g., around a remote military supply storage building. More details of the user interviews can be found in Chapter 11.

1.2 Definition of Scope

A complete UGV system performing the tasks described in the scenario above need all the capabilities listed in Figure 1.5.

A review of the current state-of-the-art robotics research around the world today reveals that these functions are at different levels of maturity, depending on what hardware is available and what degree of robustness is needed. Instead of spreading the project resources across all areas, the research efforts of the AURES project has been focused on the capabilities shown in green/solid rectangles, i.e., enhanced teleoperation, optimal positioning, optimal patrolling, communication aware surveillance, intruder tracking and search and secure.

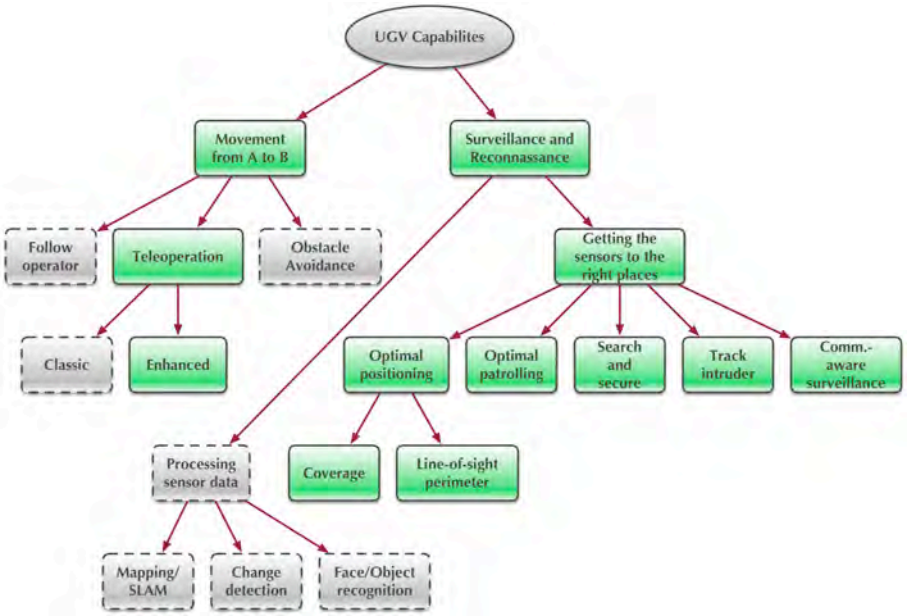


Figure 1.5: Capabilities of a complete UGV system. The capabilities drawn in solid green are the focus areas of the project.

The results of those efforts will be described below.

1.3 Scientific Contributions and Publications

The scientific contributions of the AURES project can be found in the following peer reviewed book chapters and conference papers, [5, 33, 42, 7, 34, 32, 6, 36, 26, 25, 35, 4, 27, 39, 13, 14, 15]. Technical results can also be found in the three survey reports, [24, 8, 31] and the end user interview documents [40, 41]. An overview of the titles and topics of the publications can be found in Table 1.1.

1.3.1 Report Outline

The rest of this report in organized as follows.

In Chapters 2 and 3 results regarding optimal positioning of surveillance UGVs in terms of line-of-sight perimeters and wall coverage are presented. Chapter 4 then describes the new patented enhanced UGV teleoperation control mode. Different methods for optimal patrolling are presented in Chapter 5 while Chapter 6 describes our work on the Search and secure problem. Then, Chapter 7 investigates how the character of a wireless communication channel can be taken into account to increase the effective network range in surveillance missions. Chapter 9 proceeds to presents the AURES system components in terms of robotic hardware and simulation environments. Finally, Chapter 10 describes the activities carried out during the demonstration event in April 2009 and Chapter 11 contains details from the user interviews and evaluations.

Table 1.1: Publications of Aures

Ref.	Type	Title
		Enhanced teleoperation
[36]	CDC 2007	A new control mode for teleoperated differential drive UGVs
[35]	Patent	Method for teleoperating an unmanned ground vehicle and such a vehicle
		Optimal positioning
[33]	Book chapter	Towards Optimal Positioning of Surveillance UGVs
[32]	IROS 2008	Optimal Positioning of Surveillance UGVs
[31]	Tech repor.	Survey of Positioning Algorithms for Surveillance UGVs
		Optimal patrolling
[42]	Book chapter	A comparative study of task assignment and path planning methods for multi-UGV missions
[5]	Book chapter	Minimum time multi-UGV surveillance
[8]	Tech report	Survey of patrolling algorithms for surveillance UGVs
[4]	ECC 2009	Algorithms for the Connectivity Constrained Unmanned Ground Vehicle Surveillance Problem
[6]	IFAC 2008	Communication constrained multi-UGV surveillance
[7]	CDC 2008	Cooperative surveillance missions with multiple UGVs
		Track intruder and Formation Control
[34]	IROS 2008	Improved predictability of reactive robot control using Control Lyapunov Functions
[39]	CDC 2008	Consensus under communication delays
[13]	CDC 2008	On the Stability of Distance-based Multi-Robot formations
[15]	ACC 2009	Further results on the Stability of Distance-based Multi-Robot formations
[14]	ECC 2009	Event-triggered cooperative control
		Communication-aware surveillance
[26]	RSS 2007	An Experimental Study of Exploiting Multipath Fading for Robot Communication
[25]	ICRA 2008	Communication-Aware Trajectory Tracking
[27]	IEEE Journal	Using robot mobility to exploit multipath fading
		Search and Secure
[24]	Tech report	Survey of search-and-secure algorithms for surveillance UGVs

2 Optimal Positioning of Surveillance UGVs: Creating Line-of-Sight Perimeters

This chapter contains an overview of one of the results reported in [32]. This result addresses the capability of creating a line-of-sight perimeter containing a number of buildings or objects, as illustrated in Figure 2.1 below, taken from the scenario in Chapter 1.

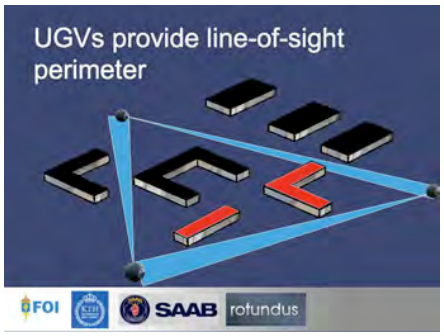


Figure 2.1: A line-of-sight perimeter, as also seen in Figure 1.2.

2.1 New Capability and Motivation

The capability of creating a line-of-sight perimeter is one important tool used to provide the operator with good situational awareness. A line-of-sight perimeter enables the operator to keep track of people entering or exiting a given area. Creating such a perimeter is furthermore something that is often done by both police and military personnel. Security personnel are in most cases to few to create a perimeter, but with the assistance of UGVs this tool becomes available to them as well.

2.2 Overview of Proposed Method

The method makes use of something called a visibility graph, shown in Figure 2.2. The visibility graph is a network of positions, where a pair of positions that are mutually visible are connected by a line. The positions are called vertices and the lines are called edges. The problem of creating a line-of-sight perimeter can now be formulated as that of picking a number of vertices in the visibility graph such that they are connected by edges enclosing the designated buildings.

For technical details about the algorithm we refer the reader to [32], but to provide some idea of how it works we give the following description. The main trick is how to manipulate the visibility graph so that we can find the optimal line-of-sight perimeter by just solving a standard shortest path problem

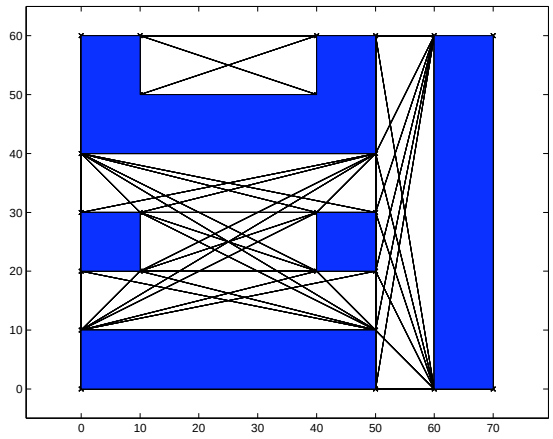


Figure 2.2: Top view of a visibility graph and a set of buildings.

Algorithm 1: The algorithm developed to find the best line-of-sight perimeter positions.

Input: A polygonal map, such as the blue/red regions in Figure 2.3.
Output: UGV positions creating a perimeter around the red regions.
Create a visibility graph G;
Pick one point p in the red region;
Pick another point q that is further away from p than any of the vertices;
Let E_p be the edges intersecting the line segment (p, q);
Remove the edges in E_p from G;
foreach *Edge* (a, b) *in* E_p **do**
  Solve the shortest path problem from a to b in G;
end
The shortest of these solutions is the desired perimeter;

If there is more than one red region an additional technical step must be taken, this is also described in [32].

2.3 Illustrating Examples

In Figure 2.3 we first provide some solutions corresponding to the map in Figure 2.2. Note that the perimeters are optimal with reference to the number of UGVs needed. A small variation described in [32] can be used to also capture the actual perimeter length in meters.

Finally, we provide a 3D example of a line of sight perimeter problem. The setting is depicted in Figure 2.4, while the solution and corresponding camera views are shown in Figure 2.5. These images conclude the chapter on line-of-sight perimeter surveillance.

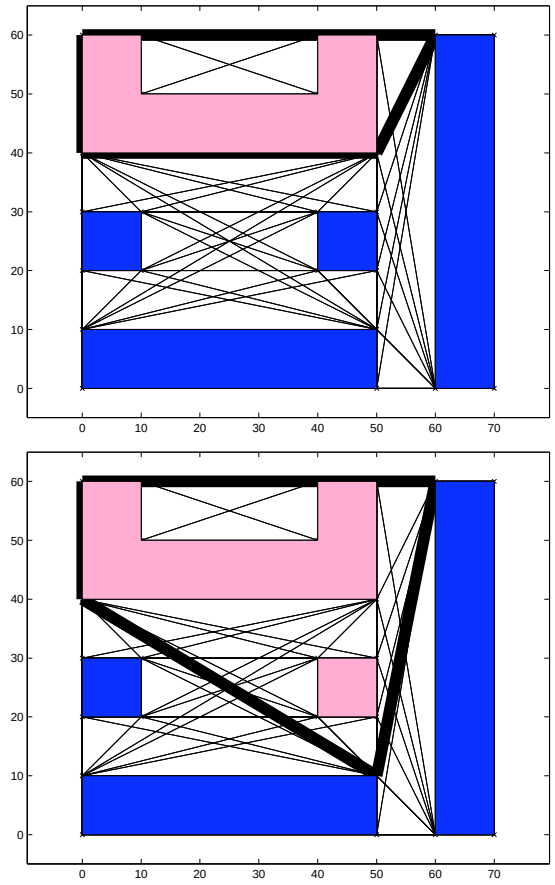


Figure 2.3: A four node line-of-sight perimeter surrounding the upper left building. Note that this perimeter can be guarded by either four one-camera UGVs, or by two two-camera UGVs such as the one depicted in Figure 1.1.

Figure 2.4: A set of buildings to be surveyed. Motion sensors have triggered alarms in both of the two buildings with darker red roofs. The line-of-sight perimeter solution can be found in Figure 2.5 while the wall coverage solution is shown in Figure 3.6.

Figure 2.5: A line-of-sight perimeter surrounding the two buildings of interest in Figure 2.4. The lower right image shows a top view of the triangular perimeter. The other three images are from the three UGV cameras.

3 Optimal Positioning of Surveillance UGVs: Creating Wall Coverage

This chapter contains an overview of results reported in [33, 32]. These results address the capability of positioning a number of UGVs such that all designated walls are covered in the camera images transmitted by the vehicles, as illustrated in Figure 3.1 below, taken from the scenario in Chapter 1.

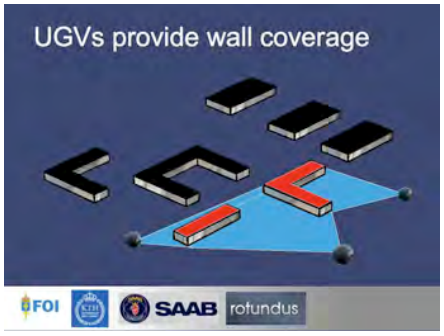


Figure 3.1: UGVs covering a set of walls, as also seen in Figure 1.2.

3.1 New Capability and Motivation

Camera equipped Unmanned Ground Vehicles (UGVs) present a flexible alternative to the numerous stationary sensors being used in security applications today. However, for such systems to be successful, efficient algorithms to compute desired camera locations that cover a set of buildings in response to an alarm, are needed.

The problem addressed in this chapter involves finding UGV positions such that all given walls are seen in the video streams from the cameras. This is to be achieved while accommodating not only zoom capabilities, but also field of view and image resolution constraints.

The problem is illustrated in Figures 3.2 and 3.3. Two buildings, and two UGV positions that cover all walls of the two buildings are seen in Figure 3.2. The solid lines indicate field of view of the UGV cameras. Notice that both buildings are inside the field of views of the two UGVs.

To get a good view of something, it is however often not enough that the object is included in the picture, but we also want the resolution of that object to be good enough. The object should not be too far away, and it should not be seen from too steep an angle. If we incorporate a somewhat stronger constraint on the resulting images the task of covering all walls can not be accomplished by 2 UGVs. Instead, a 3 UGV solution such as the one in Figure 3.3 must be found. When using computer vision algorithms, an image resolution constraint such as the above is often required to get good results.

3.2 Overview of Proposed Method

The problem above is a more constrained version of the *so-called* Art Gallery Problem, which is know to be fundamentally hard to solve (NP-complete).

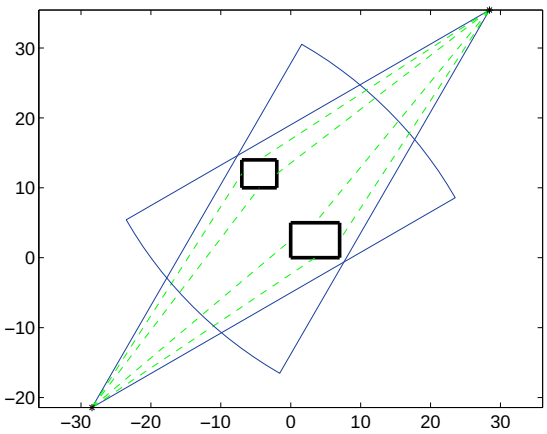


Figure 3.2: Two rectangular buildings being surveyed by two UGVs. The dashed green lines illustrates what walls are covered by what UGV while the solid blue lines denote the field of view cones.

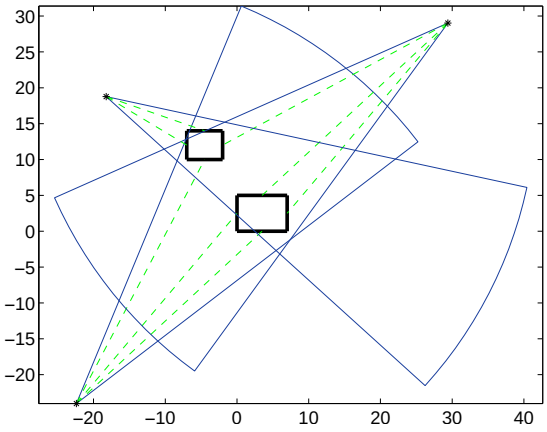


Figure 3.3: The same scenario as in Figure 3.2, but with increased resolution constraints. Note how three UGVs are needed to cover all walls.

Therefore we must settle for something less than a fast algorithms that is guaranteed to produce optimal solution. For similar problems it is common to rewrite the problems as so-called set covering problems and then apply a straight forward algorithm to find reasonably good solutions in short time. We use the same strategy, together with an elaborate trick to turn the constraints into a set of good candidate solution points to be used in the set covering problem. Somewhat more formally we get the following algorithm.

Algorithm 2: The algorithm developed to find the best wall coverage positions.

Input: A polygonal map, such as the one in Figure 3.2 and a number of walls to be covered.
Output: UGV positions covering all designated walls.
Convert all constraints to circles and line-segments;
Let the set C of candidate points be the intersections of those circles and lines;
foreach *Point* p *in* C **do**
  Compute $w(p)$, the walls guarded from this point;
end
Rewrite as a set covering problem;
Solve the set covering problem using greedy search;

The reason for using constraint intersections is twofold. First, it turns out that these constraints, that are very natural from an application point of view, result in constraint boundaries that are either circle or line segments, see Figure 3.4. Second, we can in fact show that the optimal solution can be found within these candidate points, see [32] for details.

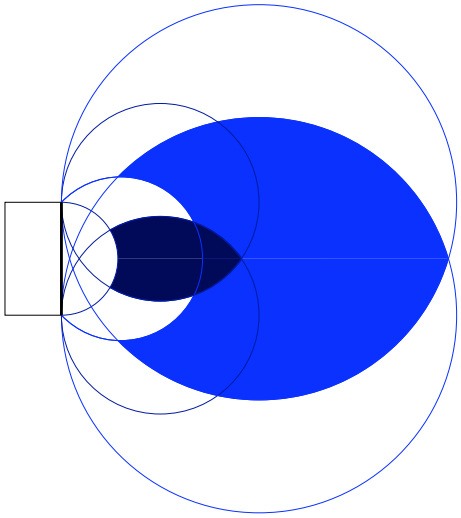


Figure 3.4: The sets satisfying both field of view and image resolution constraints for two different zoom settings. The dark smaller region corresponds to a zoom setting giving the field of view $\alpha = 90°$ and the light larger region to a zoom setting giving field of view $\alpha = 45°$.

3.3 Illustrating Example

We illustrate the approach with two examples, one quite complex in Figure 3.5, and one including the corresponding camera views in Figure 3.6. These examples concludes the chapter.

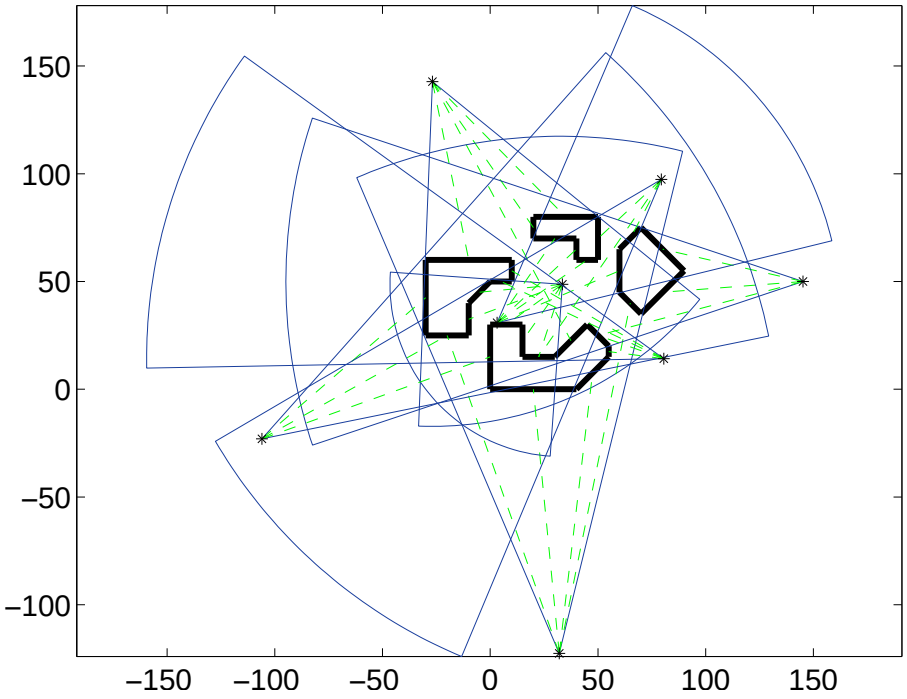


Figure 3.5: A complex scenario with 27 walls to be guarded. The solution requires 8 guards to guard all walls while satisfying occlusion, resolution and field of view constraints. As in Figure 3.3 above, asterisks (*) are guard positions, dashed green lines show what walls are guarded by whom, and blue cones illustrate field of view limitations.

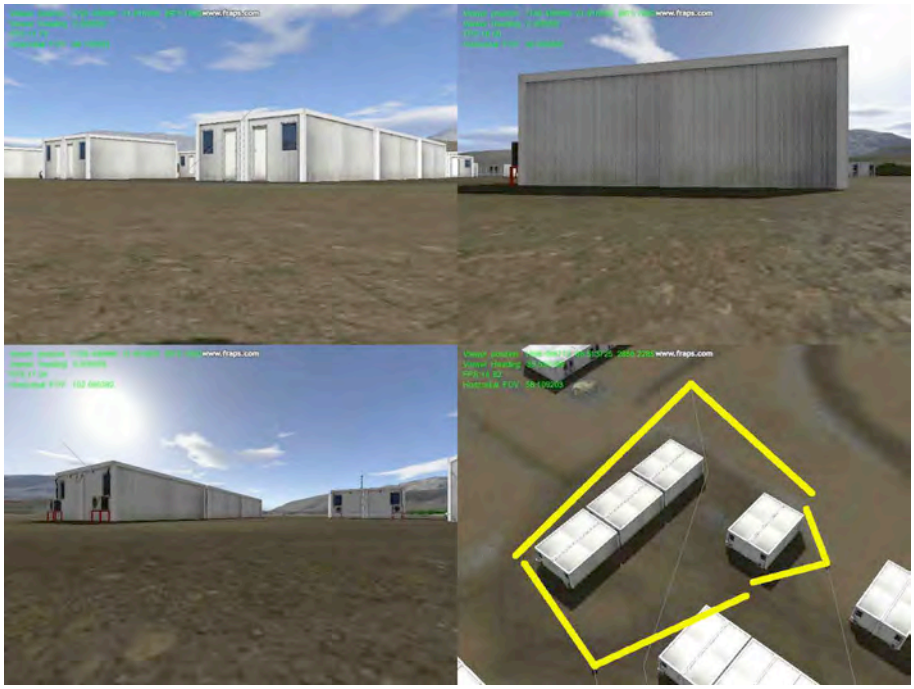


Figure 3.6: A wall coverage solution corresponding to the scenario in Figure 2.4. The lower right image shows a top view of the buildings with UGV positions and fields of view illustrated in yellow. The other images correspond to those captured by the three UGV cameras and transmitted to the UGV operator. Three different zoom settings where available to the UGVs in the algorithm. It turned out that all three were used, resulting in fields of view of 102, 58 and 90 degrees, listed in a clockwise direction starting from the lower left.

4 Enhanced UGV Teleoperation

This chapter contains an overview of the results presented in [36], that was also patented in [35]. The main result is an enhanced way of doing teleoperation, as illustrated in Figure 2.1 below taken from the scenario in Chapter 1.

Figure 4.1: The operator controls the UGV by teleoperation, as also seen in Figure 1.2.

4.1 New Capability and Motivation

Teleoperation is and will be the most important way of controlling UGVs in may application. The ideas presented below enables the operator to interact with the UGV in a new way. By introducing an intermediate control layer, a user interface that is very similar to so-called first person shooter (FPS) computer games, *e.g.* Doom and Half Life, can be created. The advantages of such interfaces is that they are intuitive, and that literally millions of potential future UGV-operators already have spent hundreds of hours training with them. The control mode gives the user direct control of the position and orientation of the on-board camera, while the actual orientation of the vehicle is abstracted away using so-called feedback linearization. The interface can be applied to both so-called differential drive robots, such as the Packbot in Figure 4.2 and to spherical robots such as the Groundbot in Figure 1.1. However, it is slightly more elegant in the former case, which is why we will use such a UGV in the examples of this chapter. At the end of the chapter, we present results of an informal evaluation using a simulated UGV, indicating that the proposed interface is indeed preferred to the classical one in some applications.

4.1.1 Detailed Motivation

Many of the most important UGV applications today, such as Explosive Ordnance Demolition (EOD) and disaster area search and rescue, are performed using teleoperation. In fact, over 20.000 EOD missions have already been carried out by US forces in Iraq and Afghanistan, [43] with robots similar to the one in Figure 4.2. Further more, in such missions performance can be vital, since operators sometimes "have only 15 minutes to work before they come under enemy fire", [12].

In EOD and search tasks, it has been noted that the system performance is often limited by the human-robot interaction, [29] and similar observations

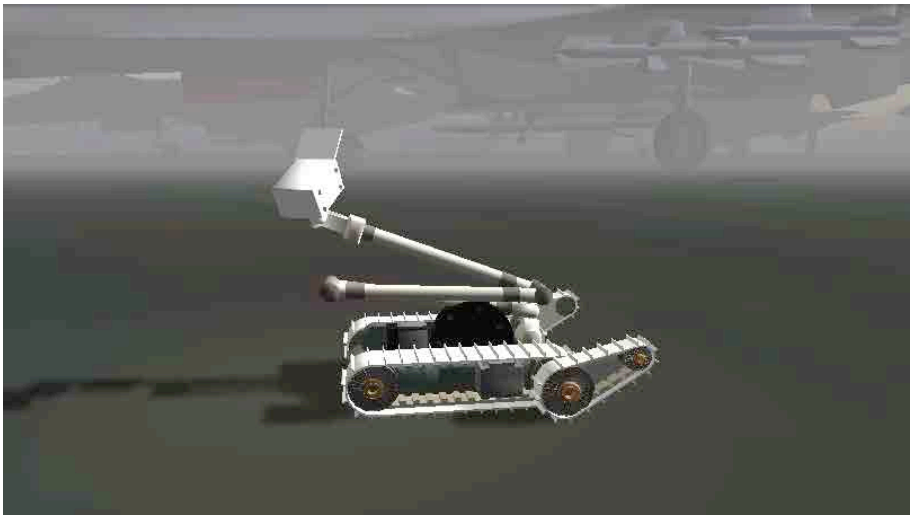


Figure 4.2: The iRobot Packbot, used by *e.g.,* US Armed Forces in Afghanistan. The snapshot is from the simulation environment used to evaluate the proposed approach, see Section 4.3.

has also been done in the military context, "In Military Operations ... rapid action is often considered a tactical necessity... The pace of robot missions is, generally, limited by the operator's ability to gain situational awareness and to control the robot, rather than by its top speed" [28]. These problems are emphasized when the robot is to move in cluttered areas near collapsed structures or inside buildings. Poor lighting conditions, as well as limitations on field of view and depth perception, makes it harder for the human operator to get the good situational awareness needed to search and navigate around obstacles in an efficient manner. In [44], Woods *et al.* describe how it is often hard for the operator to estimate scales using a video stream, leading to mistakes regarding obstacle sizes and distances. Furthermore, Burke *et al.* [11] report that operators spend as much as 49% of the mission time gaining an appropriate situational awareness. Similarly, Yanco and Drury [45] describe how understanding the robots position, relative to obstacles and landmarks, takes roughly 30% of the time. Still the operators situational awareness is often inaccurate, resulting in collisions and the robot getting stuck. One way of remedying these problems that has received a lot of attention, is to use different sensors to create a coherent world view to present to the operator, [37]. A less explored topic is that of using algorithms from the control community to improve the way the user interacts with the robot itself. A good control interface can improve the situational awareness of the operator in two ways. It can make it easier to point the camera in the desired direction during a movement, and by reducing the amount of focus needed to control the UGV it can allow the operator focus more on the surroundings of the UGV.

4.2 Overview of Proposed Method

Most UGVs today are controlled with two joysticks, one to change the UGVs position and orientation and one to control the pan-tilt unit of the cameras, and thereby control the point of regard relative the UGVs orientation. This coupling between UGV orientation and camera point of regard creates a significant control problem for the operator when navigating in confined spaces.

Figure 4.3: A gamepad with two joysticks. Control devices of this type are used for computer games and in recent times also for teleoperation control of robots such as the Irobot Packbot in Figure 4.2. This gamepad was furthermore used in the evaluations in Section 4.3.

Some robot models therefore avoid this problem by using a fixed mounted camera, while others have automated functions to either align the camera with the UGV, or align the UGV with the camera. The two last functions are steps towards freeing the operator from the burden of keeping track of the UGV orientation while watching the camera images. It is our aim to find a control mode that allows him to go one step further, and forget about it all together.

We will now first describe the classical direct robot control mode, and then the new mode that is proposed in this chapter.

4.2.1 Direct vehicle control

The standard operator interface of the UGV depicted in Figure 4.2 is a pad with two joysticks, such as the one in Figure 4.3. One joystick, say the left, is used to control the tracks to either turn or move ahead. The second, right, joystick is used to control the motors of the camera mounting thus rotating the camera relative to the rest of the UGV.

The following example illustrates how an operator would use the interface.

Assume the UGV operator is panning the camera and suddenly finds something interesting he wishes to take a closer look at. He must now carefully move the two joysticks in opposite directions in order to rotate the UGV towards the object, while the camera rotates in the opposite direction to keep the object in view. He then pushes one of the joysticks forwards to start moving towards the object. It turns out an obstacle is blocking the way. Since the scales are hard to judge, the operator is uncertain of how far away the obstacle is. To help the depth perception, he has to rotate the UGV 90 degrees, and move perpendicular to the obstacle direction while studying the camera images. Having made a number of coordinated UGV/camera rotations, separated by forward motions he manages to improve his situational awareness and navigate around the obstacle to the object of interest.

With this example in mind we now describe what a first person shooter game interface looks like.

4.2.2 FPS game control

When playing a FPS computer game, the screen shows a first person view of the surroundings of the character, see the example in Figure 4.4.

Similarly to the UGV case above, the game is controlled using a two joystick pad such as the one in Figure 4.3. The difference is that here, the right (camera) joystick controls the camera rotation relative to a world fixed coor-

Figure 4.4: Screenshot from the FPS game Half Life 2. Note the submachine gun at the bottom of the screen, always pointing in the camera direction.

dinate system, while the left joystick controls the translation of the character, relative to a camera fixed coordinate system. That is, pushing the left joystick forwards corresponds to the character moving in the direction of the camera, and pushing the left joystick leftwards means moving perpendicular to the view direction. Running the algorithm we get the results depicted in Figure 4.5.

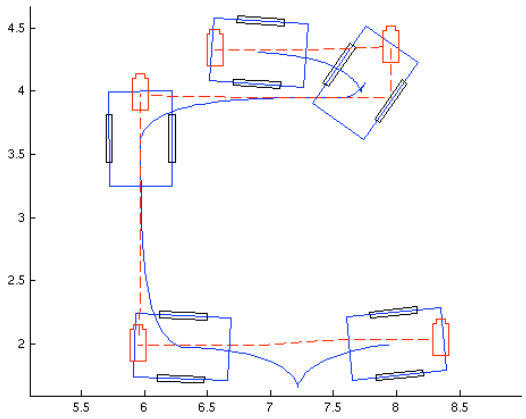


Figure 4.5: Resulting camera and vehicle trajectories when the user commands the translations left, forward, right, forward and left. Note that the camera trajectories are indeed straight line segments.

4.3 Control Mode Comparisons

To get an indication of what control mode would be preferred by potential users we performed a user evaluation where we let eight people previously unaware of the two control modes perform simulated search missions using the Packbot in Figure 4.2.

4.3.1 Evaluation method

A snapshots from the simulation environment is shown in Figure 4.6. The scenario is a runway with four fighter aircraft, and the operators were asked to search the aircraft for possible explosive devices, placed there by an enemy sabotage unit.

Figure 4.6: The simulation environment used to perform the evaluations. The small window in the upper right corner of the screen shows a view of the Packbot from behind while the main window shows what the camera sees.

To promote systematic searching, and include the situational awareness of the operator in the test, we gave the operators a schematic top view of the aircraft and asked them to mark the location of each foreign object found. We used blue markers, spheres of roughly 3 inch diameter, to denote the foreign objects and to reduce randomness in the test, we placed as many as 12 markers on each aircraft. Figure 4.7 shows one such marker.

Each operator performed four search missions, one on each of the four aircraft. During four minutes, the operators searched an aircraft, and indicated the positions of all found markers on a paper with a top view of the aircraft. For half of the operators, the first search was performed using the FPS control mode the second using the classical control mode, the third using FPS and finally the fourth using the classical mode again. The other half of the operators used the control modes in reversed order. Data was collected from the two last task, while the two first were considered as training, to remove the risk of misunderstandings regarding the task or control modes. Finally, to remove systematic errors due to how easy it was to find the markers, the order of the last two aircraft was reversed for half of the operators in each group above.

To increase the performance using the Classical control mode, we introduced a small window in the top right corner of the simulation window, see Figure 4.6. The smaller window showed the configuration of the robot, i.e., how the camera is rotated relative to the chassis. This information is often presented at UGV control stations.

The evaluations were performed on a MacBook Pro, 2.5GHz, 4GB, running

Figure 4.7: One of the twelve spherical blue markers, indicated by the white arrow, is visible at the lower left corner of the engine inlet.

a simulation produced using the 3D simulation engine Unity[1]. For both control modes the Packbot was controlled using the two sticks of the Gamepad seen in Figure 4.3.

4.3.2 Evaluation results

For each operator, we counted how many blue markers were found using the two different control modes. The operator was then given a form in which to answer some questions on a scale from 1 to 7, where 7 corresponded to *(very / considerable)* and 1 corresponded to *(not at all)*. All questions and corresponding average answers can be found in Table 4.1 below.

To determine if the differences are statistically significant, the data was furthermore analyzed using a *t-test* [19]. If the p-value computed in the t-test is lower than 0.05, the difference between the two categories is said to be statistically significant. The t-values of the table are also a measure of how different the two categories are, but does not include information on the number of tests performed. For a thorough discussion on the t-test and corresponding p and t values, see [19].

As can be seen in Table 4.1 the subjects rate the FPS mode as more favourable in most cases. FPS is significantly more intuitive ($p = 0.002$), easier to learn ($p < 0.004$), lower the mental workload ($p < 0.0008$) and is rated as working better ($p < 0.0001$). The data also shows a tendency ($p < 0.08$) of the operators to find more markers, a 15% increase in the data, when using the FPS mode. No conclusions can be drawn regarding the impact of the control modes on stress ($p < 0.22$). Finally, when being asked what control mode they preferred, one operator preferred the classical mode, one did not prefer any, and six preferred the FPS mode.

[1] See, https://www.unity3d.com (2009-04-15)

Question	Mode	Avg.	σ	t	p
How many markers	FPS	10.5	1.6		
were found?	Classical	9.1	1.3	2.0	< 0.08
How intuitive is	FPS	5.5	1.7		
this control mode?	Classical	3.8	1.3	4.8	<0.002
How hard is	FPS	2.3	1.8		
it to learn?	Classical	4.4	1.8	-4.1	<0.004
How high is the	FPS	2.6	1.1		
mental workload?	Classical	4.4	1.7	-5.6	<0.0008
Were you	FPS	3.2	1.3		
under stress?	Classical	3.9	2.0	-1.4	< 0.22
How well does	FPS	6.2	1.2		
the mode work?	Classical	4.5	1.1	10.7	<0.0001

Table 4.1: Operator Test Results

4.3.3 Discussion of results

As seen above, in the context of the evaluation, FPS is intuitive, works well, is easy to learn and lowers the mental workload. This means that the operator's resources can be used for other tasks, such as being aware of hazards and obtain a better overall situational awareness. Therefore, even if only marginally more markers were found, the advantage of using FPS in this application is substantial.

The inconclusive result on stress is propably be due to the fact that the task was not perceived as very stressful by any of the operators, and a clearer result might be found if e.g. the search time was reduced.

A clue to the benefits of using FPS in other applications can be found in the following observation. During the tests we noticed that it appeared to be hard to use the classical control mode when the camera was not pointing forwards. This was seen from the fact that many operators, including the one preferring the classical control mode, ignored the possibility of rotating the camera and performed most of the search by just controlling the robot chassis. The camera control stick was only used occasionally to change the elevation of the camera. Therefore, it seems reasonable to assume that for tasks when camera motion is not necessary, the benefits of using FPS are likely to be small, and for task where concurrent motion of camera and chassis is important, the benefits might be larger.

Note that this a small scale evaluation, indicating that the FPS control mode is worth exploring further. To get a clearer understanding of the benefits of the two control modes, more evaluations must be performed in several different areas of application. This is, however, not within the budget of the current funding.

4.4 Conclusions

Teleoperating UGVs is an important, and often difficult and demanding task. In this chapter we have presented the theoretical foundation for enabling operators to interact with differential drive UGVs in a new way. Similarly to using inverse kinematics to directly control the orientation and position of a robot arm gripper, we propose to use feedback linearization to directly control the position and orientation of a UGV mounted camera. This intermediate control

layer makes the resulting interface closely resemble those of mass market FPS computer games. Besides being intuitive, such interfaces are used daily by millions of users, performing tasks that are much more dynamic and complex than most UGV missions today. We have also presented the results of an operator test, indicating that the proposed control mode can indeed be an attractive alternative to the classical control mode in some applications.

5 Patrolling

This chapter contains an overview of the results reported in [8, 9, 42, 6, 7, 4]. These results addresses the capability of minimum time surveillance of a given area as illustrated in Figure 5.1 below, taken from the scenario in Chapter 1.

Figure 5.1: UGVs covering a set of walls, as also seen in Figure 1.2.

5.1 New Capability and Motivation

In both civil and military applications, surveillance is performed in order to assist in the prevention, detection and monitoring of intrusion, theft or other safety-related incidents. Environments that require such supervision are numerous and include airport facilities, storage buildings, harbors, factories and power plants. In the ideal case, surveillance should be performed in a continuous manner and cover the entire facility, although in practice, financial and head-count constraints limit it to only encompass the most important and critical areas.

From a performance standpoint, the potential benefits of using a group of more autonomous security or surveillance UGVs include cost savings and reduced risk exposure for human guards. Also, autonomous systems can perform many security and surveillance routines more effectively than humans since they are able to maintain a high attention level regardless of working around the clock. An additional benefit is that they do not participate in so called inside jobs.

In comparison with stationary surveillance cameras, a more mobile solution has several advantages, most apparently that of flexibility, in the sense of being able to cope with application areas that are changing in time. For instance, using stationary cameras, a great deal of camera redundancy would be required to fully monitor a harbor area where the container setup is varying on a daily basis. However, using surveillance UGVs, these changes can be easily incorporated in the planning scheme. Given the above, it is not surprising that the research area of control of surveillance vehicles is also active and growing.

5.2 Overview of Proposed Method

We have considered two different minimum time problems related to unmanned ground vehicle (UGV) surveillance. The first problem, called Mininum Time

UGV Surveillance Problem (MTUSP) is the following. Given a set of surveillance UGVs and a polyhedral area, find waypoint-paths for all UGVs such that every point of the area is visible from a point on a path and such that the time for executing the search in parallel is minimized. Here, the sensors' field of view are assumed to have a limited coverage range and be occluded by the obstacles.

The second problem, called Connectivity Constrained UGV Surveillance Problem (CUSP), extends the MTUSP by additionally requiring the induced information graph to be connected at the time instants when the UGVs perform the surveillance mission, i.e., when they gather and transmit sensor data. Below the main focus will be on the first problem formulation. The reader may consult [7, 4] to read about the proposed solution method for the second problem formulation.

The MTUSP solution method encompasses three subproblems, as illustrated in Figure 5.2. In the first subproblem, the computationally intractable problem of finding the minimum time waypoint-paths that enable *complete* regional surveillance, is turned into a finite dimensional combinatorial optimization problem. This is achieved by finding a so called *maximal convex cover* of the area, A. In the second subproblem, the order in which to visit the sets in the cover is determined using Tabu search (TS). Briefly described, TS is a metaheuristic optimization method than can escape local minimas by classifying certain search directions as *tabu*. The third subproblem, which is called as a subroutine of the second one to evaluate the objective function in the TS, involves a shortest path problem on a graph, constructed from the given visitation order.

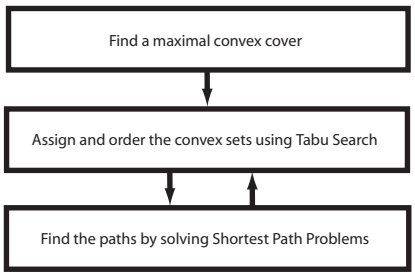


Figure 5.2: The proposed solution relies on decomposing the problem into three subproblems.

Formally we state the algorithm below.

In Equation (5.2), I_i^π is the index of the sets in C that are assigned to UGV i in the minimization of F. In (5.1), $\alpha = 1$ corresponds to the minimum time problem and $\alpha = 0$ corresponds to the minimum distance problem.

5.3 Illustrating Example

Example 5.1 *A simple example problem with one UGV is depicted in Figure 5.3 where the start position, p_1, is given while the end point is a free variable. In step 1 of Algorithm 3, the maximal convex cover $C = \{c_1, c_2, c_3, c_4\}$ is created. In step 2 the visitation ordering $\pi = (\underline{5}, 2, 3, 4, 1)$ is first tried. This permutation corresponds to the UGV (which has id number 5) visiting the convex areas in the following order: $c_2 \rightarrow c_3 \rightarrow c_4 \rightarrow c_1$. The shortest possible waypoint-path visiting the sets in this order is (p_1, p_2, p_3), and is thus returned in step 3, after minimizing $f_i(\pi)$. In the next iteration of step 2 the ordering*

Algorithm 3: Patrolling

The algorithm consists of the following two steps where the second step involves the iterative solution of two subproblems:

1. *Create a maximal convex cover $C = \{c_1, \ldots, c_M\}$ of A in accordance with Algorithm 2 in [9].*

2. *Solve the following combined assignment and ordering problem using TS:*

$$\min_{\pi} F(\pi) = \alpha \max_{i} f_i(\pi) + (1-\alpha)\Sigma_i f_i(\pi), \tag{5.1}$$

where π is a permutation of $\{1, \ldots, M+N\}$, representing the assignment/ordering of the M convex sets to the N UGVs (see Examples 5.1), $\alpha \in [0\ 1]$, and $f_i(\pi)$ is the optimal path length of UGV i given the visitation constraints dictated by π. The value of $f_i(\pi)$ is found in a sub-routine by using a shortest path formulation to solve the following optimization problem:

$$\begin{aligned} f_i(\pi) = \quad & \min_{P^i} \quad \Sigma_k ||p_k^i - p_{(k+1)}^i|| \\ & s.t. \quad P^i \text{ guards } \cup_{I_i^\pi} c_j \\ & \quad\ \ P^i \text{ visits } c_{I_i^\pi(j)} \text{ before } c_{I_i^\pi(j+1)}, j \in \mathcal{Z}^+_{|I_i^\pi|-1} \end{aligned} \tag{5.2}$$

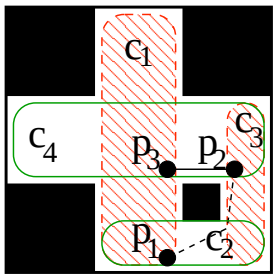


Figure 5.3: An example MTUSP problem. One UGV starts at p_1 with free endpoint. The optimal solution corresponds to the dashed line, while a suboptimal one also includes p_3, as explained in Example 5.1. The four sets of the maximal convex cover are denoted c_1, c_2, c_3, c_4.

$\pi = (\underline{5}, 1, 2, 3, 4)$ is chosen, with step 3 returning only (p_1, p_2). After some additional iterations no improvement is found and the algorithm terminates and returns (p_1, p_2).

In Figures 5.4 and 5.5, the areas to be surveyed are bounded within a polygon and the waypoint-paths of the UGVs have been highlighted. The (white) barracks and the (green) tents constitute the obstacles that occlude the onboard cameras. Figure 5.6 presents a snapshot view from the onboard cameras.

In the MATLAB simulations that follow, the initial position of the UGVs are marked with a square (■), while the final positions are marked with a diamond (♦). These two, together with the filled larger circles represent the surveillance points for guarding A. The search area, A, is chosen to be all of the obstacle free space, *i.e.*, the white area in all figures. It is assumed that

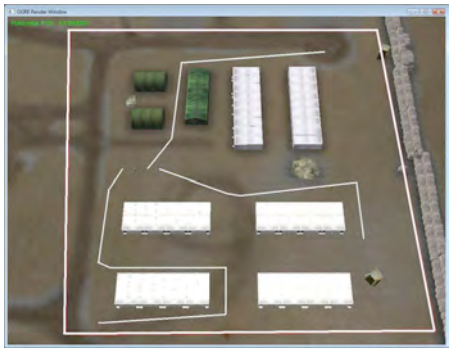


Figure 5.4: The convex cover for this particular area contains 20 sets and takes on average 1.0 seconds to generate. The last two subproblems in Algorithm 3 take on average only 92.5 milliseconds to perform. All computations have been performed on a laptop with Dual Core^{TM}, Intel®, 2.0 GHz processors.

Figure 5.5: The convex cover for this larger MTUSP instance contains 47 sets and takes on average 14.2 seconds to generate. The two last subproblems take on average 1.5 seconds to perform.

Figure 5.6: Snapshot from all four onboard cameras (enumerated row-wise).

the black obstacles have been enlarged with the diameter of the vehicle so that waypoint-paths touching an obstacle do not imply collision.

The first simulation, found in Figure 5.7, illustrates the cooperative nature of the MTUSP. The final positions of the vehicles are here *free* variables to be chosen by Algorithm 3. As can be seen, this extra degree of freedom is used constructively so that the vehicles survey the horizontally and vertically aligned "streets" in a cooperative manner with the common objective of minimizing the search time, *i.e.*, we have chosen $\alpha = 1$ in (5.1).

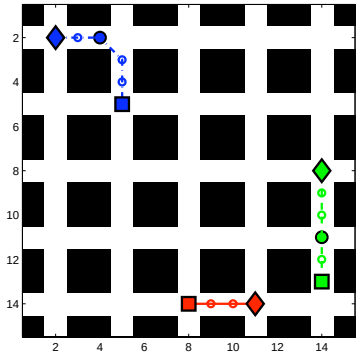
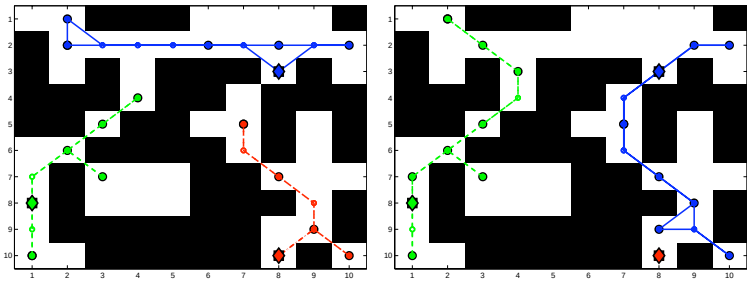


Figure 5.7: The Manhattan grid is surveyed cooperatively in minimum time. The starting points of the UGVs are marked with ■.

Figure 5.8 further illuminates the interplay between the choice of the objective function and the obtained solutions. In Figure 5.8(a), the solutions are found by minimizing the total surveillance time. It can be noted that these solutions distribute the work load quite evenly over the vehicle fleet. In Figure 5.8(b) however, the objective has been set to minimize the total distance traveled by the vehicles, *i.e.*, $\alpha = 0$ in (5.1). Since this option does not take into consideration the division of the work load between the different vehicles, the resulting solutions often do not utilize some of the vehicles at all. This may be of tactical interest when, *e.g.*, battery power must be saved, or when unemployed vehicles can be used to perform other missions in parallel.



(a) Minimum time objective. (b) Minimum distance objective.

Figure 5.8

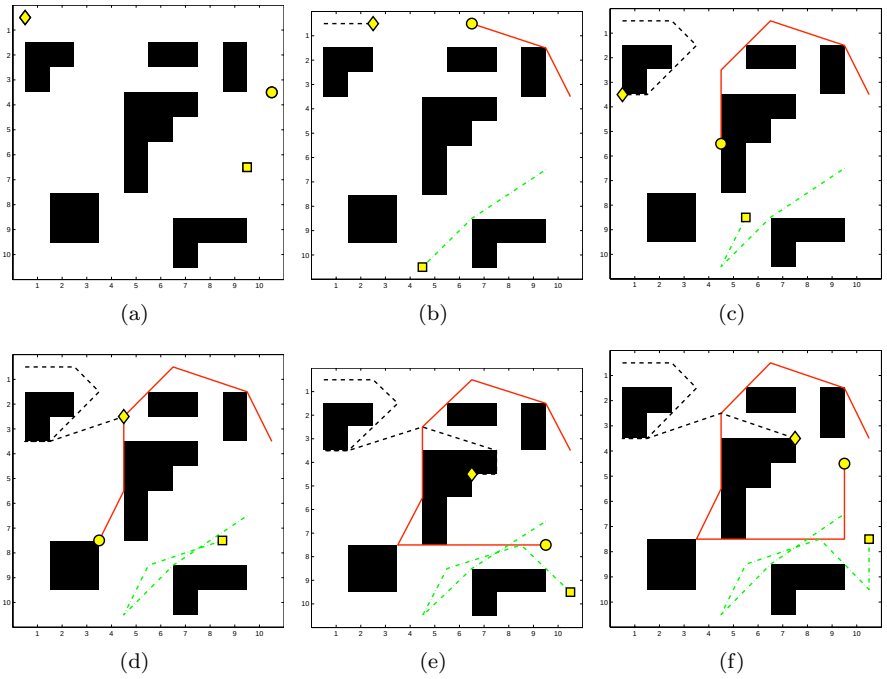


Figure 5.9: An example CUSP simulation. The most important aspect to notice is that the UGVs are not restricted to pass on the same "side" of the obstacles but are nevertheless *recurrently* connected at the five surveillance instances in Figure 5.9(b)– 5.9(f).

6 Search and secure

6.1 Introduction

This chapter contains an overview of results reported in [21]. These results address the capability of searching and securing a given area as illustrated in Figure 6.1 below, taken from the scenario in Chapter 1.

Figure 6.1: UGVs making sure that no one is hiding in the area

6.2 New Capability and Motivation

The capability of searching and securing an area refers to the ability of a team of robots to totally search a given area with buildings such that all possible intruders are detected. This includes preventing an intruder from sneaking back to a secured area without being detected.

Imagine a fleet of mobile surveillance robots, guarding an oil refinery by night. The robots patrol along randomized routes, chosen to cover the whole facility, and send camera imagery back to a manned control room. Powerful computers scan the images for intruders, fire or damages and can alert the guards if anything is out of the ordinary. If an alarm is triggered, the robots can go to the contaminated area and search it in a coordinated fashion, so that an intruder cannot get back into the cleared area undetected. This allows the guards to focus on responding to more complex situations, while these dull or potentially dangerous tasks are handled by the robots.

This project can be seen as one of these cases, meaning that the search for intruders in an area can be a dangerous and dull task for people to perform. It is desirable to explore the possibility of deploying robots, in this case UGVs, that can accomplish it. The efficiency of this choice is going to be examined and an algorithm for solving this problem and implementing this new capability is going to be proposed.

6.3 Overview of Proposed Method

The basic idea behind the algorithm is to divide the area to be searched in convex regions and abstract them to a graph. Each convex region corresponds to a node on the graph.

In order to search the area, all nodes of the graph must be covered by a UGV. In order to guarantee detection of the intruder(s) and also to prevent

the intruder(s) to sneak back into the secured area, all loops in the graph must be broken. The loops are broken by placing the so called blocking UGVs in every loop.

After the blocking UGVs have reached their positions, the searching UGVs are deployed in order to search the remaining area for intruders and thereby secure it.

The use of convex regions comes from the fact that any point within a convex region is visible by any other point inside the same region. This property implies that as long as a UGV has entered a convex region it can secure it because any intruder can be seen and therefore the intruder cannot hide from it. The difference between a convex and a non-convex region is illustrated in Figure 6.2.

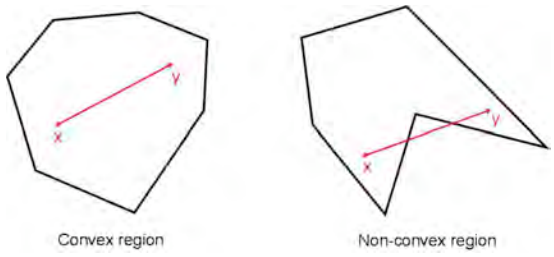


Figure 6.2: An example of the difference between a convex and a non-convex region.

The algorithm followed in order to solve the search and secure problem is as follows:

Algorithm 4: Search and Secure

Input: A polygonal map and the starting points of the UGVs.
Output: The paths that the UGVs have to follow.
Divide the camp in triangles by utilizing the Delaunay algorithm;
Merge the triangles in convex polygons;
Create the Minimum Spanning Tree (MST) from the graph of the convex polygons and find where the loops have to be broken;
Determine the actual positions of the blockers and remove the covered nodes of the MST;
Search the reduced MST in order to find the number of the needed searchers and what nodes they have to cover;
Create the paths of the blockers and the searchers;

A simple example of this algorithm can be seen in Figures 6.3 - 6.8.

An important consideration is the trade-off between the search time and the number of UGVs that will be used. This can be justified by considering the following points:

- The form of the spanning tree can be changed, meaning that the blockers could be placed in different positions and break the loops in various points. In this way the spanning tree may not be minimum but it could lead to using less UGVs. The possibility of having a tree which is not deep and has a lot of branches/leaves or a tree which is deep and has a few branches/leaves has to be explored. If for example a tree that has as few branches as possible can be created then this implies that the algorithm will produce a solution that requires less UGVs in order to be implemented (and the solution is still simple to implement since the UGVs do not have to be coordinated).

- A searching UGV can be used to search more than one branches of the MST. This would lead to a smaller number of needed UGVs but also to greater search time and greater complexity since now the searching UGVs have to be coordinated. In this case, if one UGV is used in order to search more than one branches then a second UGV is needed to block the "root" of these common branches and therefore some coordination/communication between these two UGVs is required. The algorithm can be even more complex if more than 2 UGVs have to co-operate in order to search and secure a greater area with more buildings and therefore a greater MST.
- Moving blockers could be introduced since if a blocker is used to surveil already secured areas then this blocker is no longer needed. This would also lead to a smaller number of needed UGVs but greater complexity since now the blocking UGVs have to co-operate with all the other (blocking and searching) UGVs.
- Both the above two improvements could be combined in order to both reduce (or optimize given certain criteria) the number of UGVs and the search time.

The implemented algorithm produces 2 solutions for each scenario. One solution that requires a large number of UGVs but is fast and one that requires less UGVs but is slow. The first solution requires one UGV per branch of the MST while the second solution determines and uses the minimum number of UGVs that are needed in order to search the MST.

6.4 Illustrating Example

A more complex scenario will now be presented. In this scenario an area with 9 buildings has to be searched and secured. The algorithm produces two solutions for this scenario. The first solution is faster (15 minutes and 6 seconds to search and secure the area) but requires 6 UGVs while the second one is slower (16 minutes and 45 seconds to search and secure the area) but requires 5 UGVs. Both solutions need four blocking UGVs but in the first case two searching UGVs are utilized while in the second only one. The scenario and the trajectories of the robots, as they were calculated in the first solution and simulated in the AURES simulator, can be seen in Figures 6.9 and 6.10.

Both scenarios share the common steps shown in Figures 6.11 - 6.15. The paths of the blocking UGVs for the two solutions are shown in Figures 6.16 and 6.17.

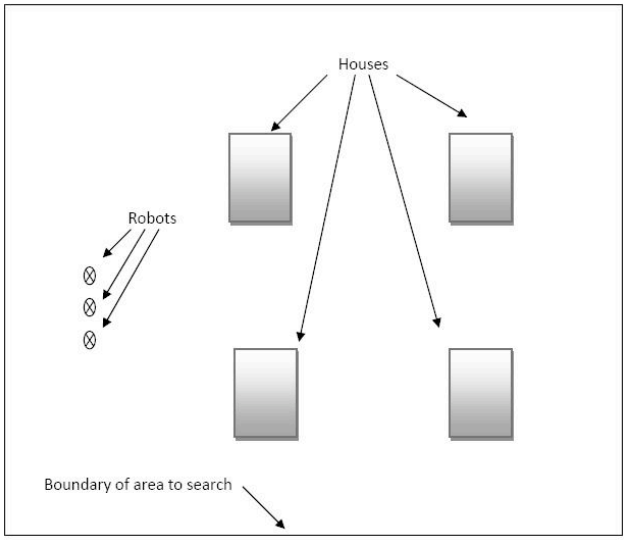


Figure 6.3: An area with 4 buildings.

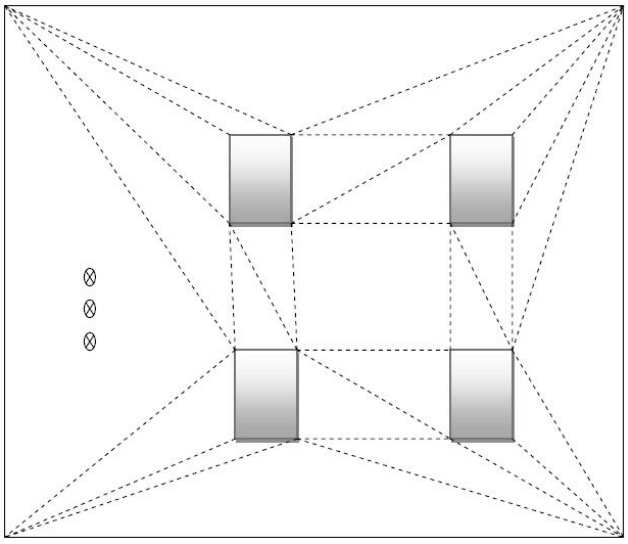


Figure 6.4: The free area is divided into triangles. These triangles will be merged in order to form convex regions.

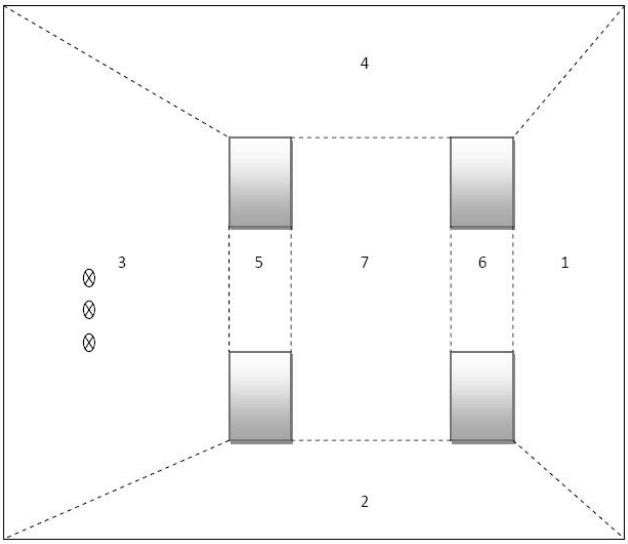


Figure 6.5: The triangles have been merged and the free area is now covered by convex regions.

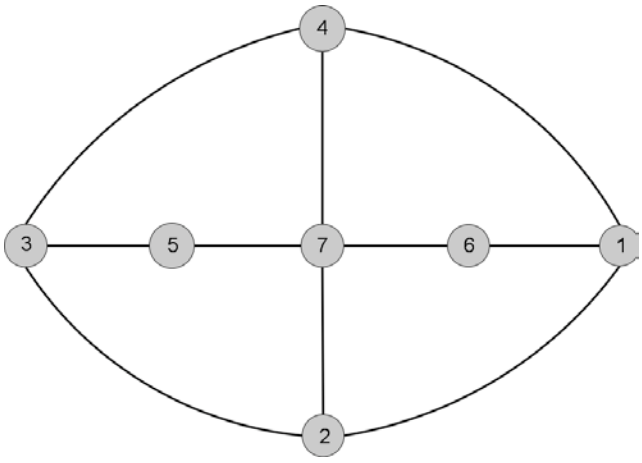


Figure 6.6: The convex regions are abstracted in the form of a graph. As it can be noticed, loops exist that must be broken in order to prevent the intruder from hiding from the searching UGVs. Each node corresponds to a convex region. Neighbouring regions are represented by edges connecting the corresponding nodes.

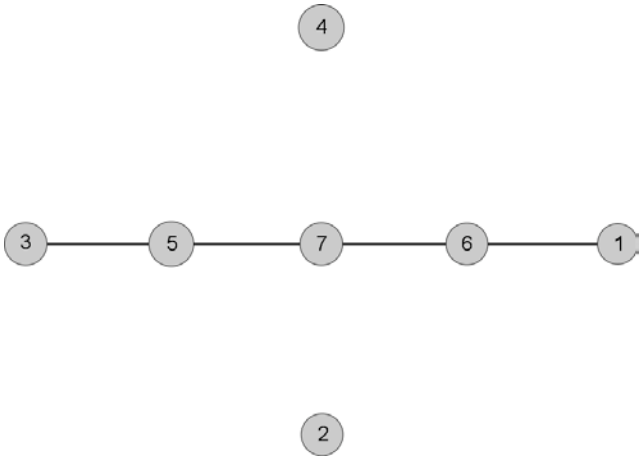


Figure 6.7: The loops in the graph have now been broken by placing two blocking UGVs at the regions 2 and 4. Since no intruders can move through egions 2 or 4 without being detected, all edges to nodes 2 and 4 can be removed. This creates the reduced MST, consisting of nodes 3, 5, 7, 6 and 1. One searching UGV is now needed to search the remaining nodes 3, 5, 7, 6 and 1.

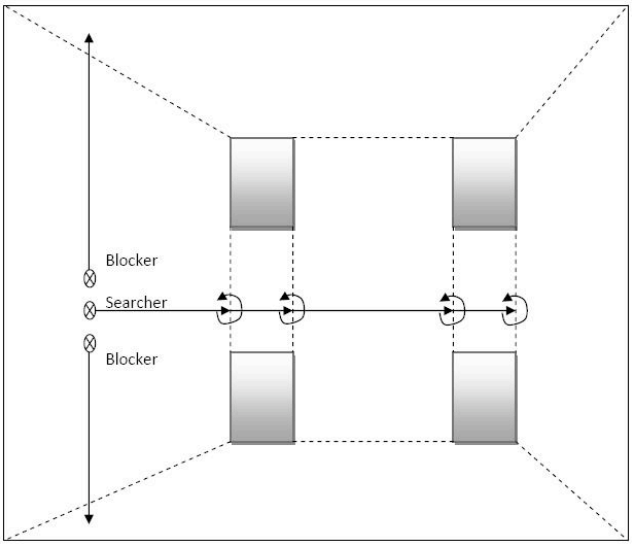


Figure 6.8: The actual trajectories of the UGVs can be seen. After the blockers have reached their final positions, the searcher is deployed in order to search the remaining regions and secure the area.

Figure 6.9: The scenario is shown in the AURES simulator environment. The trajectories of the blocking UGVs are illustrated as white line segments on the ground.

Figure 6.10: The trajectories of the blocking UGVs have now been added as white line segments. As it can be seen the blocking UGVs have reached their positions before the searching UGVs start searching for intruders.

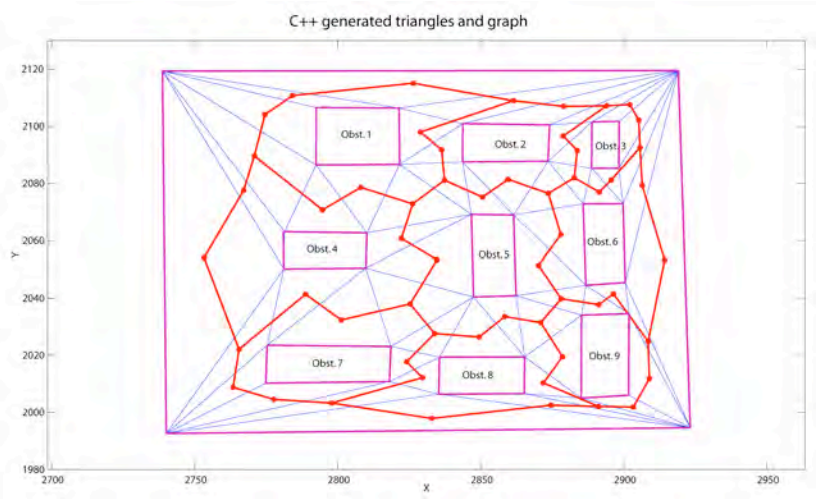


Figure 6.11: The free area is divided in triangles (blue line segments). Their centroids form the triangle graph (red line segments).

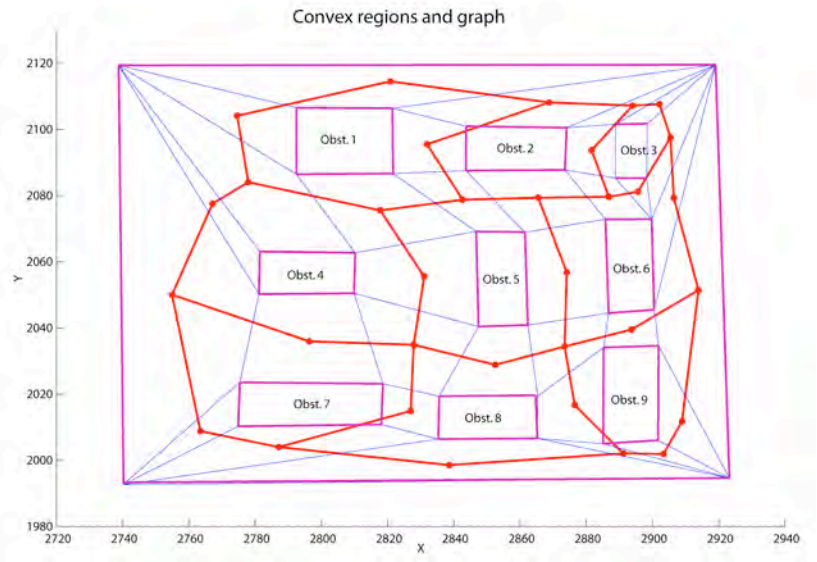


Figure 6.12: The triangles are merged in order to form convex regions (blue line segments). Their centroids form the region graph (red line segments).

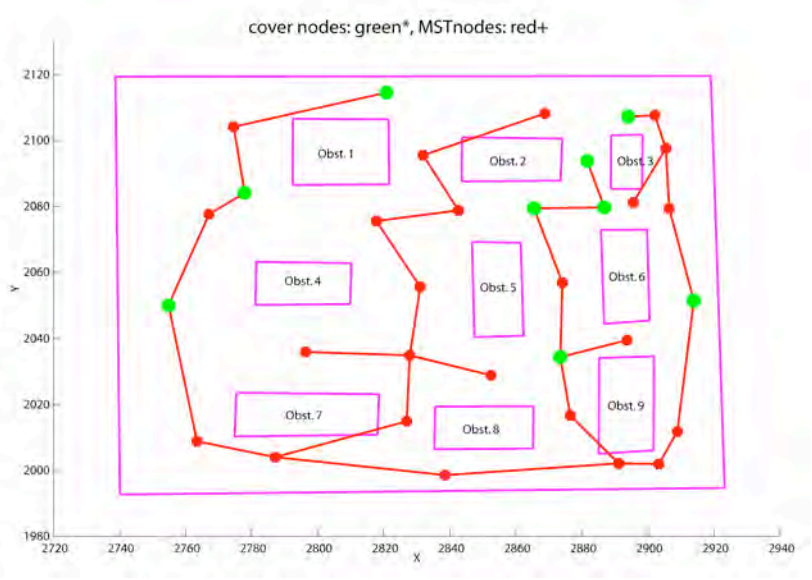


Figure 6.13: The MST (red line segments) is created and the candidate positions of the blockers are found (green points).

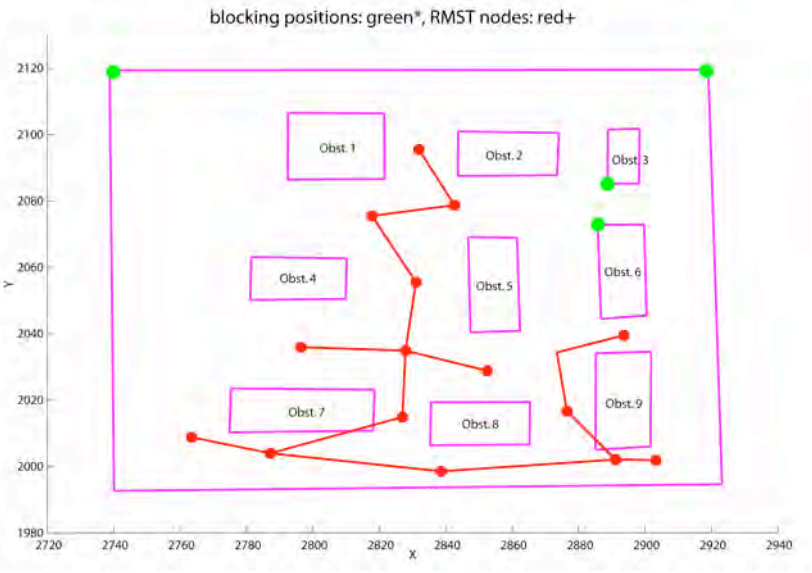


Figure 6.14: The actual positions of the blockers (green points) are calculated and the reduced MST is formed (red line segments).

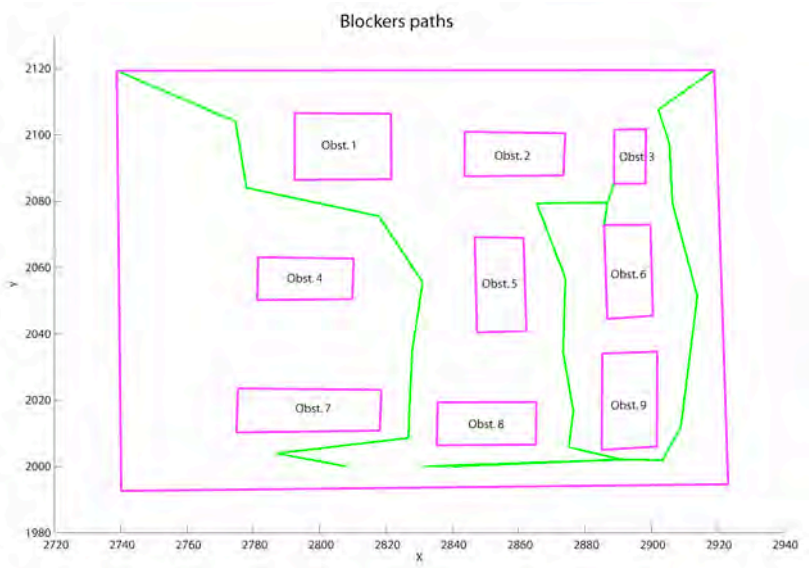


Figure 6.15: The blocking UGVs paths (green line segments) are determined.

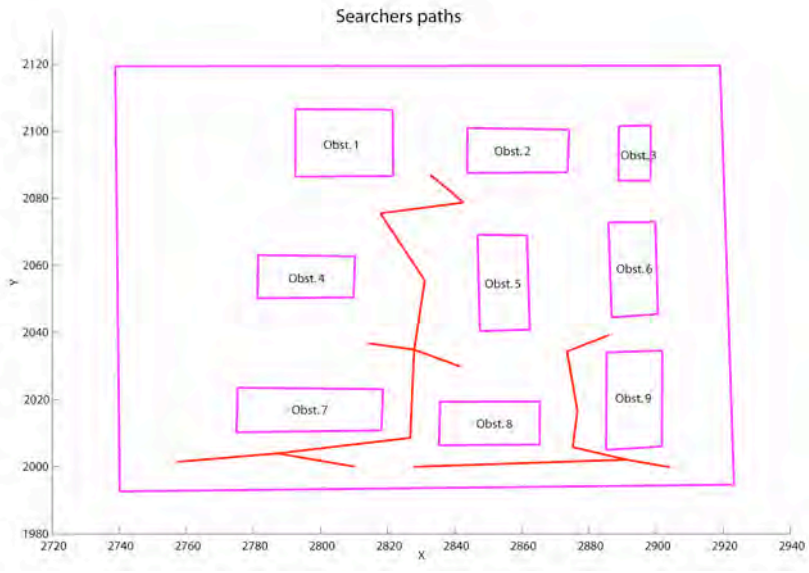


Figure 6.16: The paths of the searching UGVs for the faster solution. Two UGVs are deployed in order to search the two branches of the MST.

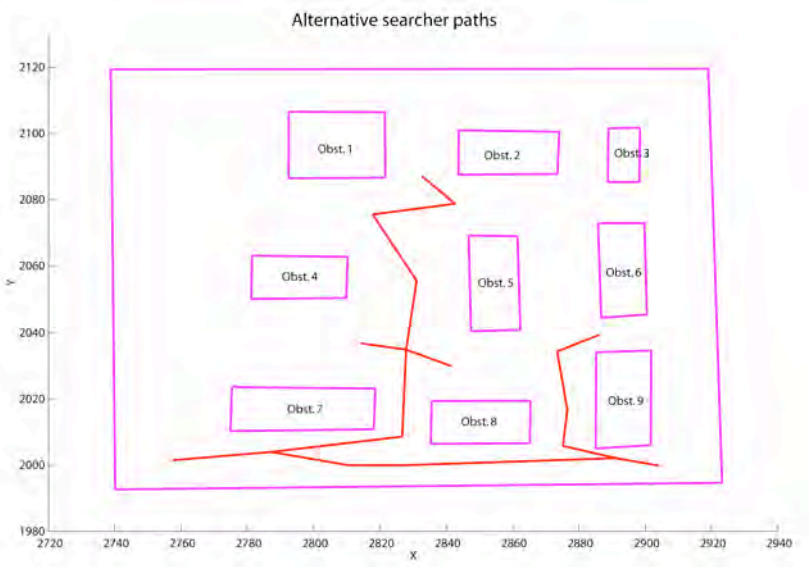


Figure 6.17: The paths of the searching UGVs for the slower solution. One UGV is deployed in order to search both branches of the MST. It first searches and secures the left branch and then the right branch.

7 Communication Constrained Surveillance

7.1 Introduction

This chapter contains an overview of results reported in [23, 25, 26]. These results address the capability of communication-aware motion control which allows mobile networked robots to increase the average communication throughput. We exploit that robots can measure the signal-to-noise ratio (SNR) in a communications channel and adapt their motion to spend slightly more time at positions where the channel is good. Two new such strategies are analyzed and evaluated: periodic stopping, where the stop duration is a function of the SNR, and controlled stopping, where the robot stops when the communication buffer is filling up. It is shown that the expected average channel capacity can be twice as high as when no information is utilized. Experimental evaluation of the strategies confirms the theoretical results.

7.2 New Capability and Motivation

A system like the one described in Section 6.2 employs advanced communications and robotics. To enable coordinated control, and collection of sensor data, the robots need communication links that provide high quality of service (QoS). Important metrics are throughput, network delay and outage probability. With cameras becoming smaller and cheaper by the day, nodes in the network can be expected to deliver high bandwidth information with low delay tolerance. Achieving this in a mobile robot network poses several challenges that must be handled, including the following: First, the nodes are resource-constrained, both in terms of energy and computation power. This requires methods of scheduling sleep for sensors and transceivers, as well as cross-layer design of medium access (MAC) protocols that can adaptively trade QoS for energy [17]. Second, since the nodes are moving and are often spread out geographically, ad-hoc routing mechanisms are required that can adapt to changing network topologies [2]. Third, to improve the reliability of delivery without too many retransmissions that cost power and cause varying delays, new transport layer protocols must be developed [3].

Coordination strategies for multi-robot systems are often formulated as decentralized control laws. Information is in these systems only exchanged between neighboring robots, so local controllers have to be designed to converge to some global behavior despite the limited communication [30]. Recently, there has been a growing interest in the robotics community to study such distributed control problems under the QoS constraints described above. An example closely related to our motivating scenario is to make the robots map the environment while cooperatively searching it in minimal time, without getting too far apart [10]. A coordination problem with another type of communication constraint is considered in this paper, but first we briefly review some common models of communication in mobile robotics.

7.3 Overview of Proposed Method

7.3.1 Models and system architecture

To illustrate how multipath fading can be exploited, we will use the scenario in Figure 7.1: A group of robots is used for surveillance of an office floor during night. The robots patrol the offices along given paths and stream camera images to a base station, which in turn feeds the data to an operator. Robots far from the base station use multihop relaying to be able to cover all rooms. The exact motion timing is not important, as long as the sensing objective is fulfilled: to provide image data from all rooms within a given time. It is, however, crucial that the robots maintain low latency and high throughput for the link to the base station. Otherwise, the video images will be noisy and intruders will have time to escape before being detected.

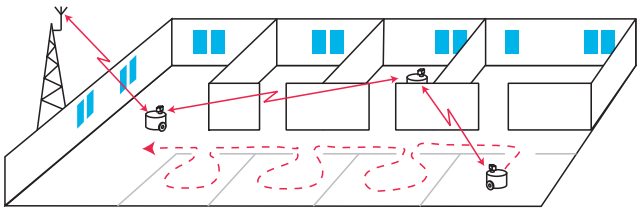


Figure 7.1: Our example scenario: a group of robots patrolling an office floor. Each robot is streaming video to a base station, either directly or by relaying through another robot. The robots need to adapt their motion to maintain high radio throughput.

Since the sensing objective provides this degree of freedom, each robot can modify its motion to improve communications. In an environment with multipath fading, this could mean spending slightly more time at positions that offer low channel attenuation and quickly passing points where the channel is worse. Note, however, that the task must still be completed before a given deadline. Since the fading varies over distances of a wavelength, finding such positions only requires small deviations along the trajectory. We will present and analyze methods for doing this tradeoff between communication and tracking, under different assumptions on what feedback the robot gets from its radio. To simplify the presentation, we will consider the case of a single robot communicating with a base station. The strategies presented here for point-to-point communication provide the basic functionality needed by higher-layer protocols for maintaining connectivity within a whole group of robots.

In the following sections, we state our model of the robot and reduce it to one-dimensional motion along the reference trajectory. We also introduce a channel model of static Rayleigh fading. The model is validated through measurements in our lab. Finally, we define the link capacity as the byte reception rate, which will be used to compare different motion strategies. In the end and the system architecture is defined.

7.3.1.1 Robot Model

The position of the robot is $q \in \mathbb{R}^2$. We assume that it has a pre-planned time-stamped reference trajectory $q_{\text{ref}}(t)$, moving at a velocity $v_{\text{ref}}(t)$, and a controller for following it. This allows us to reduce the problem to considering the one-dimensional motion of the robot along the reference trajectory. Let Δ

be the position of the robot along the trajectory, relative to the reference, so that $\Delta > 0$ means that the robot is going ahead of the reference. Also let φ be the relative velocity. Stopping the robot can be done by applying breaks or short-circuiting the motors, which does not consume battery power. We model the motion control as a hybrid system, where the robot can be in one of two modes, $\sigma = \texttt{stop}$ or $\sigma = \texttt{drive}$. The dynamics of the one-dimensional motion are then

$$\sigma = \texttt{stop}: \begin{cases} \dot{\Delta} = \varphi \\ \dot{\varphi} = -k_v(\varphi + v_{\text{ref}}) \end{cases} \quad \sigma = \texttt{drive}: \begin{cases} \dot{\Delta} = \varphi \\ \dot{\varphi} = a, \end{cases}$$

where the controls are $u = (a, \sigma)$ and $k_v \gg 1$ is chosen to model the robot stopping quickly.

7.3.1.2 Channel Model

We assume a Rayleigh fading environment where the SNR, γ, is exponentially distributed with average γ_0. We consider trajectories where the distance to the receiver does not change significantly and there is constant shadowing, so γ_0 does not change over time. In a deployed system, the approach presented here could therefore be complemented by other components that avoid shadowing from obstacles and adapt the large-scale motion of the robot to limit the path loss.

Since multipath fading is caused by multiple reflections of the signal against objects, one can expect that if nothing in the environment moves, the resulting fading should not change over time, but only as a function of the position of the transmitter and receiver. Successive minima occur about every half wavelength [20]. This is a reasonable assumption in applications such as nighttime surveillance, rescue missions in collapsed buildings or military exploration of possibly hostile environments. As described in the experiment section, we have validated the model by measurements which gave the histogram in Figure 7.2. The figure also shows the ideal Rayleigh distribution function, which fits the measurements well. By measuring the change in SNR over time when the transmitter and receiver were not moving, it was confirmed that the fading does not change in a static environment.

To allow comparison between different control strategies, we define the normalized link capacity $c(\gamma)$ as the probability of correct reception of one byte when the SNR is γ. Other choices could be to study the bit error probability or packet reception rate for packets of several bytes, but we believe that the byte capacity is an illustrative measure of the link performance. It also does not need assumptions on protocol issues such as packet size, error-correcting codes or retransmission schemes, which can be used to improve the performance on a packet level.

7.3.1.3 System Architecture

The overall system is comprised of the robot platform and the radio. As illustrated in Figure 7.3, the position q of the platform determines the SNR, γ, and channel capacity, c. A buffer in the radio stores data arriving at a rate r from a sensor. The size of the buffer is $z \geq 0$ and its dynamics are

$$\dot{z} = r - c.$$

We will first consider the case when the controller has no feedback from the radio or buffer, and thus follows the reference trajectory without stopping.

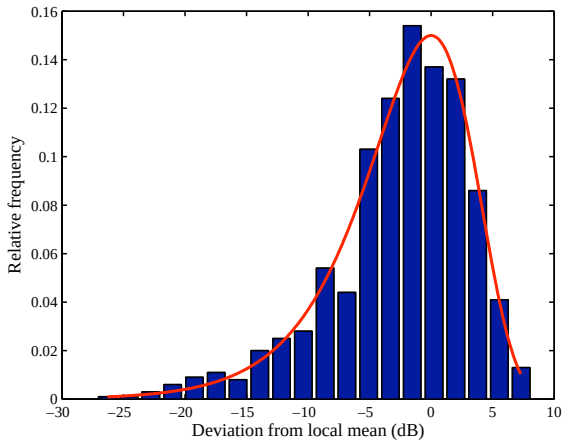


Figure 7.2: Normalized histogram of 1000 samples, taken 1 cm apart in our lab. The probability distribution function of Rayleigh fading (in dB) is included as a reference.

Then we will study the periodic stopping strategy, where the loop is closed between the radio channel and the motion of the platform. It is assumed that γ can be sampled only when standing still, as is the case in many slowly sampling radio transceivers. Finally, we will consider controlled stopping, where the controller has access to continuous measurements of both γ and z, so it can stop when needed and find local maxima of γ.

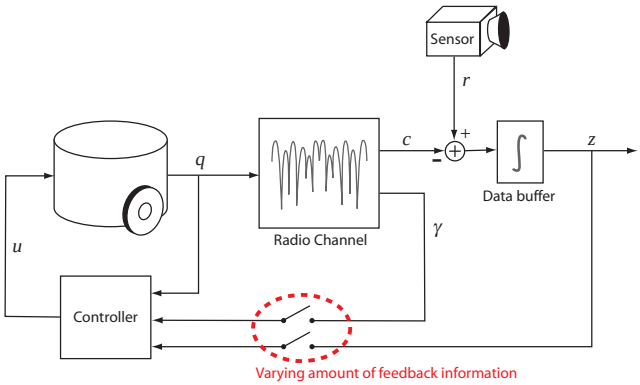


Figure 7.3: Architecture of the system, with a robot platform, a radio and a controller. The radio buffers data from a sensor and sends it through a wireless channel, whose capacity depends on the position of the robot. We present strategies for the controller with and without feedback on the SNR γ and the state z of the data buffer.

7.3.2 Communication-Aware Motion Control Strategies

In the following Subsubsections the various motion control strategies are presented and a comparison among them is made. First, no stopping. Second, periodic stopping where the SNR can be sampled when standing still. Third, controlled stopping, using continuous measurement of the SNR and the amount

of data waiting to be transmitted. For each strategy, we analyze the expected link capacity.

7.3.2.1 No Stopping

With no feedback from the radio, we assume that the robot drives along the desired path without adjusting its motion to the radio channel. The control law is

$$u = u(q).$$

The nominal link capacity as a function of the average SNR can be described as $c_{\text{drive}} = E\{c(\gamma)|\sigma = \texttt{drive}\}$. We will use this as a baseline, to compare against the more advanced motion strategies below.

7.3.2.2 Periodic Stopping

If the radio hardware needs some time to sample the capacity, the robot has to stand still to avoid the channel changing. A possible approach is then to schedule periodic stops and use the measured SNR to determine the length of the stop. The control law is now

$$u = u(q, \gamma).$$

We suggest the following strategy: The robot drives at velocity $2v_{\text{ref}}$ for a constant time, τ_{drive}. Then it stops, measures the SNR and determines the length τ_{stop} of the stop. After waiting τ_{stop}, it starts driving again. To get the desired average velocity, we require that the expected stop time is equal to the drive time $E\{\tau_{\text{stop}}(\gamma)\} = \tau_{\text{drive}}$.

We have investigated two candidates for the function $\tau_{\text{stop}}(\gamma)$: a linear policy and a threshold policy. The linear policy can be expressed as

$$\tau_{\text{stop}}(\gamma) = \max\{0, \lambda(\gamma - \gamma_0)\},$$

which achieves the desired average velocity if $\lambda = \tau_{\text{drive}}\gamma_0^{-1}e$, where e is the base of the natural logarithm.

The threshold policy is to have a constant stopping time and stop if γ is higher than some threshold value γ_{th}:

$$\tau_{\text{stop}}(\gamma) = \begin{cases} \alpha\tau_{\text{drive}} & \text{if } \gamma > \gamma_{th} \\ 0 & \text{else,} \end{cases}$$

where $\alpha > 1$. The choice $\gamma_{th} = \gamma_0 \ln \alpha$ ensures that we get the desired average velocity. Increasing the parameter α means making fewer but longer stops, which increases the resulting link capacity at the expense of larger deviations from the reference trajectory. The extreme policy $\alpha \to \infty$ corresponds to making a single very long stop at the point where the signal strength is the highest. However, high values of α make the strategy very sensitive to errors in the channel model, so we have found $\alpha = 4$ to work well in our experiments. The expected resulting link capacity for each policy will be computed in the comparison below and typical trajectories will be illustrated in the experiments.

7.3.2.3 Controlled Stopping

If the controller measures the SNR and the amount of buffered sensor data continuously, it can choose to stop the robot at local maxima of the link capacity whenever it needs to communicate with a higher capacity than c_{drive}. But

stopping also carries a cost in that the robot falls behind the reference position. This suggests an adaptive strategy that makes the robot stop to communicate when the buffer starts filling up and then makes it drive to catch up again with the reference. The control law can be written as

$$u = u(q, \gamma, z).$$

Since local maxima of the capacity are less than a wavelength apart, we assume that it takes negligible time to find a point where the capacity is greater than or equal to some value c_{stop}. For the problem to be feasible, we also assume $c_{\text{drive}} < r < c_{\text{stop}}$. Under these assumptions, the two modes of the robot described earlier also control the dynamics of the data buffer: Either it drives and communicates with link capacity c_{drive}, or it stands still and communicates with a higher capacity c_{stop}. When driving, the tracking error decreases but the motors consume power and the buffer fills up. When stopping, the buffer size can decrease but then the robot is falling behind the reference so the tracking error grows.

We use dynamic programming to compute a feedback controller that simultaneously maintains low tracking error, buffer size and power consumption. Based on the robot position and buffer size, the controller dictates the mode of the system and, when in the `drive` mode, also the acceleration [25].

The controller drives the system towards periodically switching between `drive` and `stop`. The controller adapts the duty cycle of the switching to balance the outflow and inflow of the buffer. While doing this, it also makes a tradeoff between deviation from the reference position and power consumption, which affects the switching frequency. The resulting link capacity will be computed below, and an example trajectory will be shown in the experiment section.

7.3.2.4 Comparison

The presented strategies above, no stopping, periodic stopping and controlled stopping, are examples of the tradeoff between communication performance and reference tracking. Reference tracking imposes timing constraints on the motion, so the robot can only stop and communicate long enough for it to be able to catch up with the reference afterwards. Power is also a concern, since stopping often or for long times requires more energy for catching up. To assist an application developer in making the proper tradeoff, the expected link capacities for each strategy are illustrated in Figure 7.4, as a function of the average SNR, γ_0. We have assumed non-coherent frequency shift keying, with a bit error rate of $\frac{1}{2}e^{-\frac{\gamma}{2}}$, so

$$c(\gamma) = \left(1 - \frac{1}{2}e^{-\frac{\gamma}{2}}\right)^8.$$

Periodic stopping with a threshold policy is illustrated for $\alpha = 4$, when the fixed stop time is four times as long as the drive time. As expected, the more complex controller for stopping on demand can achieve the highest capacity, at the expense of reference tracking and locomotion power. The graph also illustrates how all strategies using feedback from the radio give capacity improvements in the transition region where the signal is getting weaker, but make no difference if the signal is very strong or very weak. For example, the periodic stopping strategy with a linear stop time policy gives improvements of over 100% compared to constant motion in the interval -6 dB$< \gamma_0 <$4 dB.

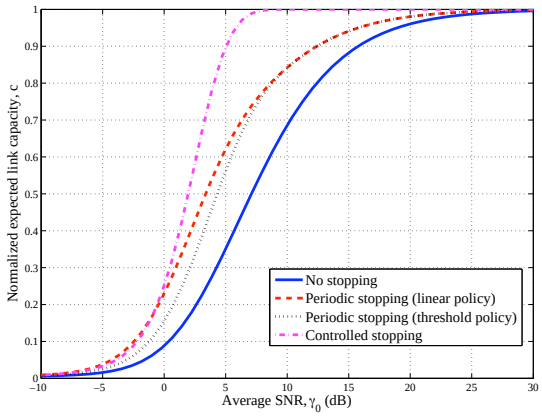


Figure 7.4: Expected normalized link capacity for the strategy of periodic stopping, with either linear or threshold stop time policies, and for controlled stopping. The strategy of no stopping is included as a reference. The curve for controlled stopping is an asymptotic upper bound since the controller adapts the capacity to what is needed to keep the buffer size bounded.

The graphs for periodic stopping can be derived by first computing the expected amount of data transmitted per stop, which is $E\{c(\gamma)\tau_{\text{stop}}(\gamma)\}$. This gives an expected average link capacity of

$$E\{c(\gamma)\} = \frac{E\{c(\gamma)\tau_{\text{stop}}(\gamma)\} + c_{\text{drive}}\tau_{\text{drive}}}{2\tau_{\text{drive}}}.$$

For the strategy of controlled stopping, the controller can switch between two link capacities: c_{drive} and a higher c_{stop}. It adaptively sacrifices reference tracking and locomotion power to adapt the average link capacity to the buffer inflow, so if the control signal is unlimited, the asymptotic maximum average capacity is c_{stop}. This capacity is achieved by the controller finding local maxima of the SNR and stopping there. Based on experience from measurements, we assume that these local maxima correspond to the 90th percentile of the capacity, which is the level plotted in Figure 7.4.

7.4 Illustrating Example

To evaluate the strategies presented above under realistic channel conditions, we have used a robot to measure the actual signal strength fluctuations as a function of position in our lab. Each strategy has then been simulated, using the measured sequence to compute the channel capacity along the trajectory. As described in the modeling subsection, we have also used the collected data to validate the model of static Rayleigh fading.

7.4.1 Experimental Setup

Our measurement robot has unicycle kinematics and a laptop onboard for control. To the laptop was connected a TMote Sky sensor node, equipped with a IEEE 802.15.4 compliant CC2420 2.4 GHz transceiver. The CC2420 has a detector for received signal strength (RSS) with a stated accuracy of 6 dB, but our experience is that the relative accuracy of the detector is in the same range as the resolution, which is 1 dB. Another TMote Sky was used as test

transmitter, sending 64 50-byte packets per second at a data rate of 250 kbit/s. The transmitter was placed on a height of about 2 m in one end of the lab and the robot was placed in the other end, with the receiver at a height of about 0.3 cm. There was no direct signal path, since the lab is full of computers and other metal equipment that effectively scatters the signal. The lab was unoccupied during the measurements, to mimic the conditions in an office at night.

To record the signal strength, the robot was driven 1 cm at a time, then stopped and recorded the number of received packets for 1 s. At each position, the average received signal strength (RSS) of the packets was recorded. Due to lack of space, the robot drove along a straight line, stopped after 2.5 m and was turned manually in place to follow a new line. We recorded 1000 samples with an average RSS of -67 dBm. Figure 7.2 shows a histogram of how the RSS measurements varied around this constant average.

Finally, we also tested the assumption on the fading being static over time, by moving the transmitter to different locations and measuring the RSS over 3 min with the robot standing still. Then the RSS was registered separately for each packet, with no averaging. The standard deviation was 1.1 dB or less for all measurement series. No outliers were more than 2.7 dB from the average.

7.4.2 Results

Figure 7.5 shows the simulation results for periodic stopping with a linear stop time policy (left) and a threshold policy (middle), as well as for controlled stopping (right). For the threshold policy, we used $\alpha = 4$. It is assumed that the reference position is moving along the x-axis at 0.1 m/s and for each control strategy, we have plotted the position q_x of the robot, as well as the reference position (dashed). The buffer size z is also illustrated, for an inflow of $r = 0.6$.

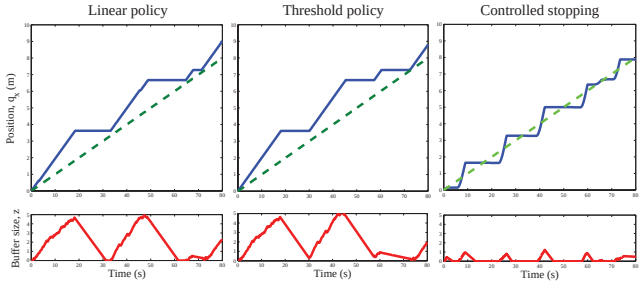


Figure 7.5: Resulting trajectories when simulating the proposed motion strategies, using channel properties recorded by measurements. The dashed lines represent the reference position, moving at constant velocity, and the lower curves illustrate the buffer size. Periodic stopping with a linear stop time policy (left) or a threshold policy (middle) give larger buffer sizes and worse reference tracking than the controlled stopping strategy (right).

Periodic stopping with a linear stop time policy gave a 71% improvement in average link capacity compared to no stopping. The corresponding improvement for periodic stopping with a threshold policy was 69%. The strategy of controlled stopping gave a link capacity equal to the inflow, since the buffer was almost empty at the end of the simulation. This means a 67% improvement over no stopping. Figure 7.5 also illustrates how controlled stopping actively keeps the buffer size low and that it results in better reference tracking than the periodic stopping strategies. It can be noted that, for a FIFO buffer, the

size of the buffer is a measure of the data latency. So controlled stopping also achieves the lowest data latency.

Controlled stopping assumes that a high signal strength can be found immediately when stopping. To simulate this, whenever the controller decided to switch to the `stop` mode, the robot sampled the channel every 0.1 s and was forced to stay in the `drive` mode until it found a position where $c \geq r$. At some parts of the trajectory the SNR was lower, which caused the robot to overshoot the reference trajectory some when driving in search of a good enough position. Guided by Figure 7.4, we used the parameters $c_{\text{drive}} = 0.36$ and $c_{\text{stop}} = 0.9$.

To reduce the influence of possible interference, the measurements were performed using the highest possible transmission power, 0 dBm. But, as commented above, stopping strategies give the best result when the link is on the limit of losing contact. To better illustrate this, we have assumed a high noise level so that the average SNR for the simulation becomes 5 dB.

7.4.3 Conclusions

We have analyzed and evaluated methods to improve the capacity of wireless robot communication, in environments that exhibit multipath fading. The main idea is to make it stop and communicate at positions where the channel is better, while still respecting timing constraints posed by tasks such as sensing. Two main strategies were considered: Periodic stopping and controlled stopping. These strategies make different assumptions on the information available to the controller from the radio and also yield controllers of different complexity.

Theoretical analysis, assuming Rayleigh fading, shows that both strategies can give significant improvements of the channel capacity compared to no stopping, using no feedback from the radio. The more complex strategy, controlled stopping, can achieve the highest improvement by adaptively sacrificing reference tracking. It is also important to note that these methods contribute the most in the transition region where the channel capacity starts to decay, but make no difference if the signal is very strong or very weak. Simulations, using actual channel properties, show that the strategies work also under more realistic conditions. The periodic stopping strategy appears to be more robust to errors in the channel model than the controlled stopping strategy. An interesting direction of future research is to employ feedback to adapt to changes in the average SNR. This can happen when moving over longer distances where shadowing and path loss can vary.

This method could be applied also in high bandwidth systems such as video links, where the fading may be frequency selective. The receiver could then be equipped with an equalizer and the SNR after equalization could be fed back to the motion controller. The resulting SNR may not be Rayleigh distributed, but the statistical analysis could be adapted to the new distribution.

We end by noting that the methods presented here could be combined with other approaches to mitigate multipath fading, such as antenna or frequency diversity. Antenna diversity is achieved by placing multiple antennas far enough apart for them to experience uncorrelated fading. That closely parallels the presented methods, where instead a single antenna is moved between sampling instances. Depending on the available mounting space for antennas on the robot and the tracking error that can be tolerated, antenna diversity and communication-aware motion therefore complement each other to improve communication performance.

8 Track intruder and formation control

8.1 Introduction

This chapter overviews the methodologies developed within the TAIS-AURES project for the case of formation control and are reported in [13, 39]. We first present a method for distance based formation control. More realistic communication scenarios are considered in the next sections where we consider communication time-delays and event-triggered sampling in our cooperative control scenarios. The previously mentioned methodologies are illustrated in Figure 8.1 taken from Chapter 1.

Figure 8.1: The detected intruder is tracked by the UGVs. At the same time the UGVs maintain a formation.

8.2 New Capability and Motivation

The problem considered in this Chapter is multi-agent formation control, where agents usually represent multiple robots or vehicles that aim to converge to a specified formation. The desired formation can be either static or moving with constant velocity. Maintaining a formation while moving is crucial for tracking an intruder. This capability of the UGVs will give the opportunity to the security officer to know the position and the actions of the intruder and possibly get an image of the intruder.

Among the vast literature on formation control, two main approaches can be distinguished: position-based and distance-based formation control. In the first case, agents aim to converge to desired relative position vectors with respect to a subset of the rest of the team. Although position-based formation stabilization is a well studied topic, there appears to be a lack of relevant results for the case of distance-based formations. Motivated by this lack of control laws, we pursue in Subsection 8.3.1 the problem of distance-based formation control.

A more realistic scenario involves time-delays in the model. In particular, each agent is assumed to have access to the information of its own state with no delays, but can only consider delayed information of the states of its neighbors. This scenario will be studied in Subsection 8.3.2.

8.3 Overview of Proposed Method

8.3.1 Distance-based Formation Control

In this section we propose a control law that is based on the distance of an agent from its neighbouring agents. The results are first presented for for two types of agents with different kinematic models. The discussion that follows is based on [13],[15].

8.3.1.1 System and Problem Statement

We consider a group of N kinematic agents operating in an area. Each agent can only access some of the points of this area due to its kinematic model. Moreover, each agent has assigned a particular orientation. The objective of the control design is distance-based formation control.

Each agent can only communicate with some of the other agents. The desired formation can be encoded in terms of an undirected graph, from now on called the *formation graph* , whose set of vertices is indexed by the team members, and whose set of edges contains pairs of vertices that represent inter-agent formation specifications. Each edge is assigned a positive parameter, representing the distance at which the two agents should converge to. The problem is to derive control laws, for which the information available for each agent is encoded in the set of its neighbours, that drive the agents to the desired formation.

8.3.1.2 Preliminaries

A *path* of length r from a vertex i to a vertex j is a sequence of $r+1$ distinct vertices starting with i and ending with j such that consecutive vertices are adjacent. For $i = j$, this path is called a *cycle*.

If there is a path between any two vertices of a graph, then the graph is called *connected*. A connected graph is called a *tree* if it contains no cycles. A *spanning tree* in a connected graph is a tree sub-graph that contains all the vertices of G. An *orientation* on the graph G is the assignment of a direction to each edge. A graph is called oriented if it is equipped with a particular orientation.

If a graph contains cycles, then its *cycle space* is the cycles in this graph. The edges of each cycle in the graph have a direction, where each edge is directed towards its successor according to the cyclic order.

8.3.1.3 Control strategy for agents with single integrator dynamics

We provide first in this section the control strategy for agents with single integrator dynamics. Assume that agents' motion obeys the single integrator model:

$$\dot{q}_i = u_i, i \in \mathcal{N} = \{1, \ldots, N\} \tag{8.1}$$

where u_i denotes the velocity (control input) for each agent and N is the number of agents.

In [13] it is proven that there exists such a control law for a required formation if and only if the formation graph is a tree.

8.3.1.4 Control strategy for nonholonomic agents

In this section we modify the control design of the previous section in order to tackle with agents that have nonholonomic kinematic unicycle dynamics. The control law used in [16] for agreement of multiple nonholonomic agents is redefined in this case to treat distance based formation stabilization. Agent motion is now described by the following nonholonomic kinematics:

$$\begin{aligned} \dot{x}_i &= u_i \cos\theta_i \\ \dot{y}_i &= u_i \sin\theta_i \quad , i \in \mathcal{N} = \{1, \ldots, N\}, \\ \dot{\theta}_i &= \omega_i \end{aligned} \tag{8.2}$$

where u_i, ω_i denote the translational and rotational velocity of agent i, respectively.

It is proven that a control law exists [15] and this control law forces the nonholonomic multi-agent system to behave in exactly the same way as in the single integrator case.

8.3.2 Consensus under Communication Delays

In this Subsection we examine a particular case of the consensus problem when the information exchange between the communicating agents has inherit time-delays. The delays of the proposed controller model various phenomena of networked systems such as transmission delays on the transfer of data between each agent and its neighbors, packet losses in wireless communication networks and inaccurate sensor measurements. Moreover, delays can result from sampling. It has been observed that a sampled signal can be seen as a delayed signal with a particular delay and specific properties. The proofs are found in [39].

We first review the original non delayed consensus problem for N agents with fixed but non necessarily symmetric communication links. Then we modify the control law for that problem to cope with the inherent delays found in our system. The open-loop dynamics are given by:

$$\dot{x}_i(t) = u_i(t), \quad i \in \{1, \ldots, N\}. \tag{8.3}$$

Each agent has access to the information concerning its state without any delay. However the data coming from the other agents are received after a time-delay caused by the various reasons given in the introduction. Consider further as an approximation that all the communication delays are constant and equal to τ which can be assimilated as an average delay. We then derive a control law [39]. This control law is proved to drive all the agents to a stable formation.

8.3.3 Event-triggered Cooperative Control

An important issue that arises in the implementation of distributed algorithms is the realization of the communication and control actuation schemes. In that respect, a futuristic multi-agent system design may equip each agent with a small embedded micro-processor, which will be responsible for collecting information from neighboring nodes and actuating the control updates of the individual agent, according to some ruling. Scheduling of these actuation or execution times can be done in a time-driven or an event-driven fashion. The first case involves the traditional approach of sampling at pre-specified time instances, usually separated by a specific period. Since the microprocessors

are assumed to be resource-limited, an event-triggered approach seems more favorable. In addition, a proper design can also preserve desired properties of the system, such as stability and convergence. In this section we consider both the cases of centralized and decentralized event-triggered control. In the first case, it is assumed that their exists a global embedded microprocessor that collects information about the whole system and triggers the feedback events for each agent. We show that there exists a lower bound on the inter-event times, i.e., the time between two consecutive actuation updates. The decentralized case is treated then where now each agent is equipped with its own embedded microprocessor that can gather only neighboring information. Similar yet more conservative results are obtained. The following is based on [14].

8.3.3.1 Control strategies

Consider a system of N kinematic agents operating in an area. We assume first that agents' motion obeys the single integrator model (8.1).

We furthermore assume that the control law can be actuated only at discrete instances of time instead of acting continuously. In contrast to traditional sampling approaches, in this paper it is assumed that the control law is actuated at instants triggered by events. In the case treated first, the control scheme is centralized and it is assumed that their exists a global microprocessor that collects information about the whole system and triggers the control actuation events for the whole team. This will be relaxed in the decentralized case.

The proposed control law in the centralized case is defined as the event-triggered analog of the ideal control law.

At each event time the control law is updated and remains constant until a new event happens. Once the control task is executed the error is reset to zero.

In the decentralized case, each agent updates its own control input at event times it decides based on information from its adjacent agents. Hence, each agent takes into account the last update value of each of its neighbors in its control law. The control law for each agent is updated both at its own event times, as well as at the event times of its neighbors.

8.4 Illustrating Example

Some examples on the distance-based formation design are provided here. In the first simulation we provide a comparison of the single integrator and nonholonomic unicycle cases. We first consider an example where the control law fails to stabilize a system of three single integrator agents to a desired triangular formation. The graph considered is a complete cycle graph. The agents start from initial positions $q_1(0) = [0, 0]$, $q_2(0) = [-1, 0]$ and $q_3(0) = [1, 0]$. The evolution in the single integrator case is depicted in Figure 8.2, where the crosses represent the initial positions of the agents and their final locations are noted by a black circle. The system converges to an undesired steady state given by $q_1 = [0, 0]$, $q_2 = [-0.6866, 0]$ and $q_3 = [0.6866, 0]$. The exact same initial positions are used in Figure 8.3, where we now consider nonholonomic agents. As witnessed in the figure, the agents in the nonholonomic case converge to the desired triangular formation. The difference is due to the nonholonomic constraints in the agents' motion in the second case.

The next example involves four single integrator agents. In the first example we have a a complete graph and a rectangular formation, to which the agents do indeed converge, as depicted in Figure 8.4. By deleting the edges between agents 1,3 and 2,4 the resulting equilibria are now shown in Figure 8.5. In fact,

in this example, agents 2 and 4 converge to the same point, since there is no edge and hence no repulsion between them.

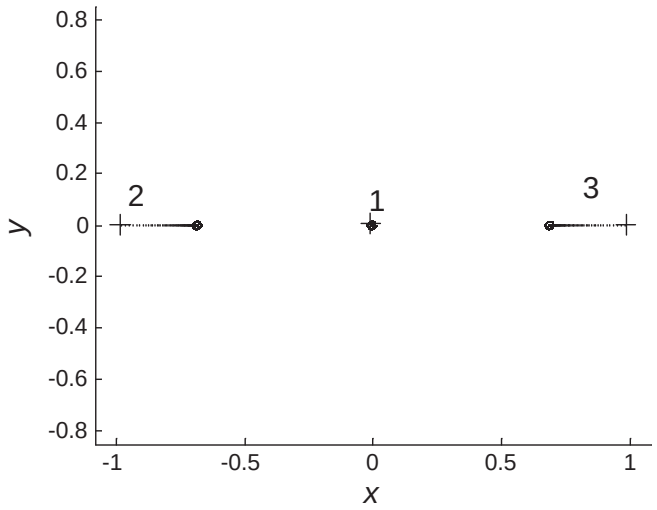


Figure 8.2: Example of three single integrator agents. The resulting configuration belongs to the cycle space of the graph.

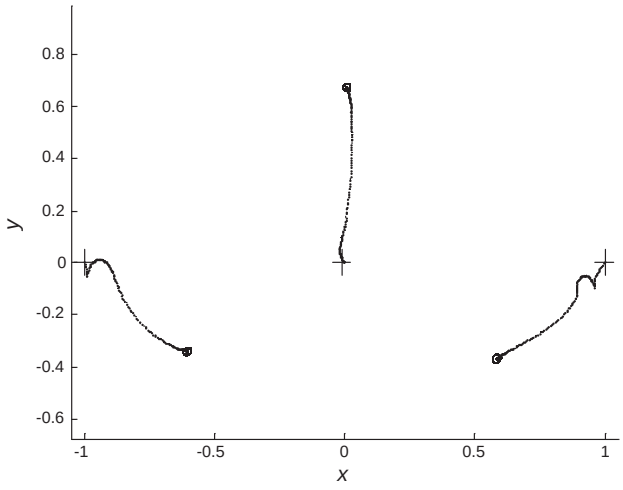


Figure 8.3: The three agents now have nonholonomic kinematics. The system converges to the desired final formation from the same initial conditions as in the single integrator case.

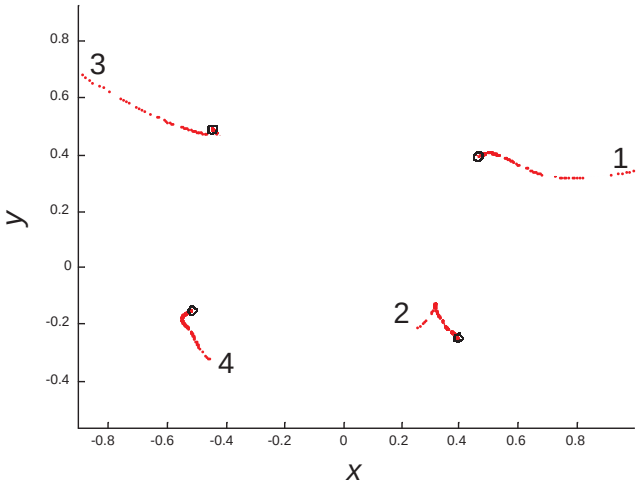


Figure 8.4: Four agents and a complete formation graph reach a rectangular formation.

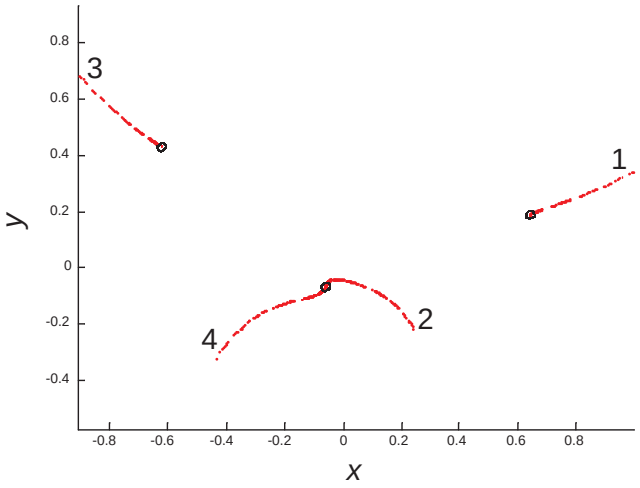


Figure 8.5: The edges between agents 1,3 and 2,4 are deleted. The agents end up in a different equilibrium point than the previous case. In fact, agents 2 and 4 converge to the same point, since there is no edge and hence no repulsion between them.

9 Hardware and software of the AURES system

In this chapter we will describe the hardware and software used to demonstrate the capabilities developed in AURES.

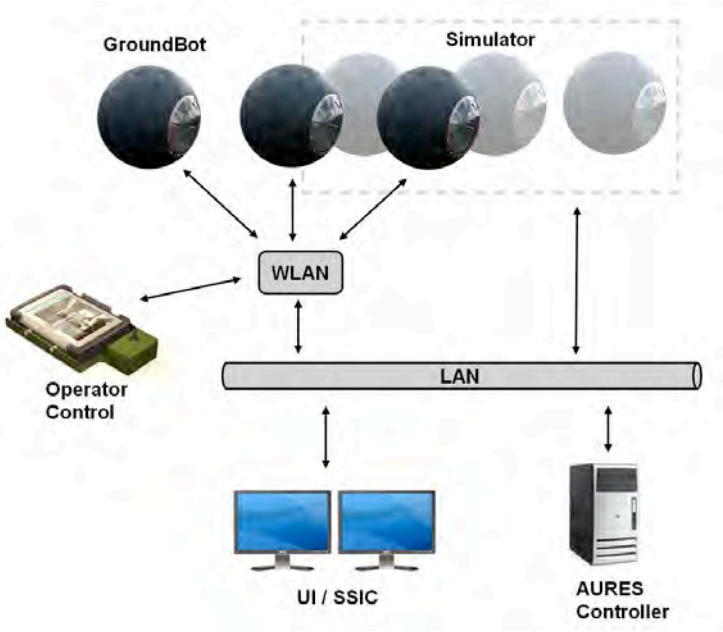
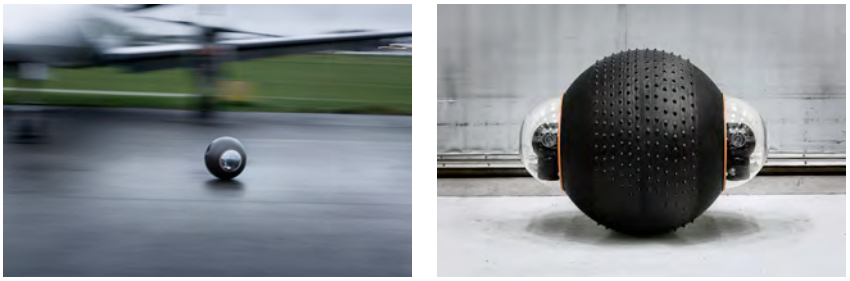


Figure 9.1: The high-level design of the demonstration testbed. The algorithms run on a separate computer and communicates with the UGVs over a network. This design allows us to work in a mixed HW/SW setting where some of the UGVs only exist in the simulation environment while others are physical robots operating in the real world.

9.1 Overview

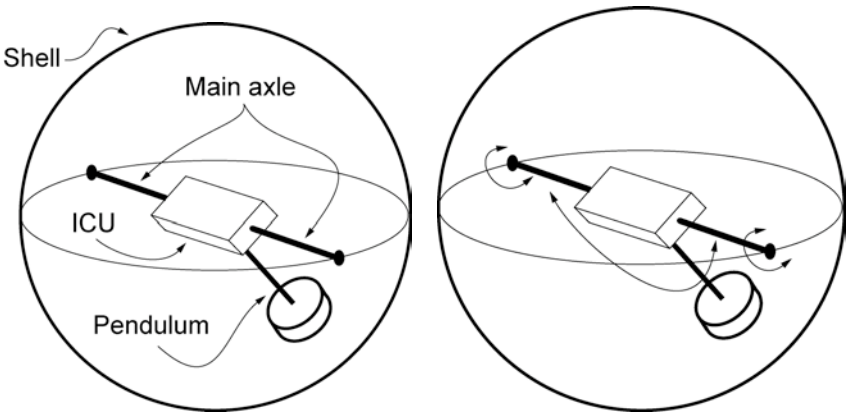
The complete system is illustrated in Figure 9.1 and consists of the following main components:

- Physical UGVs, the Groundbot.
- A simulator for simulation of one or several UGVs.
- AURESnet, the communication API that connects the components.
- Operator Control, OP, a device to manually control one UGV at time.
- A User Interface, UI, mainly for control and presentation of UGVs in a map.
- AURES Controller, the intelligence in the multi-UGV system that produces high-level commands.



(a) GroundBot at an airport. (b) GroundBot inside a hangar.

Figure 9.2: Exterior of the Rotundus UGV GroundBot.



(a) Principal parts of the UGV. (b) Principal motion pattern of the locomotion mechanism.

Figure 9.3: Principle of GroundBot's locomotion.

The main function of the UGVs is to support the operator with video from the scene. The UGVs are therefore equipped with onboard daylight vision sensors that have controllable zoom and orientation. In order to perform evaluation of the algorithms without physical UGVs, a simulator can be used. The simulator produces video from a 3D-environment. Mixed demonstrations with both real and simulated UGVs can also be performed in order to evaluate scenarios that include more vehicles than are physical available. Below we will describe each of the components in more detail.

9.2 UGVs: The GroundBot

In this section the UGV itself will be described in some detail.

9.2.1 Principle of motion

GroundBot is a spherical robot platform developed by Rotundus AB.[1] The exterior, shown in Figure 9.2, couldn't be simpler: a ball. But behind the facade, GroundBot is humming with advanced technology. That's one of the reasons why it's as easy to use as an arcade driving game.

The secret to its efficiency is the simple and robust, patented drive mecha-

[1] http://www.rotundus.se/ (2009-04-15)

Figure 9.4: An early GroundBot prototype in snow.

nism, shown in Figure 9.3. A controlled pendulum keeps the center of gravity very close to the inside surface of the sphere. At rest, the center of gravity is close to the ground. To create momentum, a built-in motor raises the pendulum. This changes the center of gravity and GroundBot begins to roll in the direction of the pendulum movement. It is a system that is as simple as it is efficient. GroundBot can move backwards and forwards. Acceleration and deceleration are fast and smooth. Move the pendulum to the side, and GroundBot turns. Not only does it take very little energy to drive GroundBot, but virtually no noise is generated either. So GroundBot moves around without drawing attention to itself.

9.2.2 Main features

The spherical design is simplicity itself. With its large circumference, GroundBot takes most kinds of terrain in its stride. And yet its appearance is friendly and unthreatening, which can be an important feature.

Inside, all cameras and sensors are sealed off from the outside world. This means that they are protected from bangs and knocks. In addition, the outside world is protected from potential electrical sparks, e.g., when GroundBot is investigating gas leaks. GroundBot moves through mud, sand and snow without getting stuck, as is shown in Figure 9.4. GroundBot is surprisingly light. Conventional surveillance robots, tipping the scales at over 200 kg, have a tendency to get bogged down in soft, unresisting surfaces. GroundBot weighs just 25 kg and can roll across all kinds of surfaces with ease.

The other reason why GroundBot can handle all kinds of terrain is its sheer size. GroundBot, with its 60 cm diameter, is slightly larger than a standard automobile tire. This means it just rolls over uneven surfaces taking them in its stride. And because GroundBot is sealed and has such a low density it can even float. There's nothing sticking out that can get caught, damaged or broken off since all cameras and sensors are safely stowed away inside the sphere. In addition to this, the entire system is hermetically sealed from the outside world. This has the following advantages:

1. Sand cannot get inside GroundBot and cause problems in the moving parts.

2. Water cannot get inside GroundBot.

3. GroundBot can be used to investigate suspected gas leaks, since the gas cannot come into contact with electrical sparks generated by the robot's

motor.

4. GroundBot can be used as a surveillance UGV in a harbor area, e.g., near oil tankers, without risk to generate potentially devastating electrical sparks.

GroundBot is also tough. It's designed to take knocks and drops, and of course, it doesn't have problems with overturning. It is possible to develop custom built versions of GroundBot which would handle drops from up to 4-5 meters hight.

9.2.3 General specifications

Terrain capabilities

- Operates in most terrain including deep snow, ice, mud and sand.
- GroundBot also floats on water.

Navigation and endurance

- Speeds of up to 10 km/h (6 mph)
- Fast acceleration and fast stopping
- Turns in a small radius left or right
- Direct control via joystick / direct user interface
- Route-following outdoors and indoors via waypoint navigation system

Power

- Operating time 8-16 hours depending on mission profile
- Battery Li-Ion rechargeable
- Charging time 3-4 hours

Communication

- Wireless connection with Wi-Fi as standard
- Emergency/power control over a separate secure radio link

Video

- High quality video streams in MPEG-4 compression
- Sent via Real-time Transport Protocol (RTP)

Payload

- Standard payload: two pan-tilt-zoom cameras, giving 360° field of vision
- Payload capacity is 2 kg

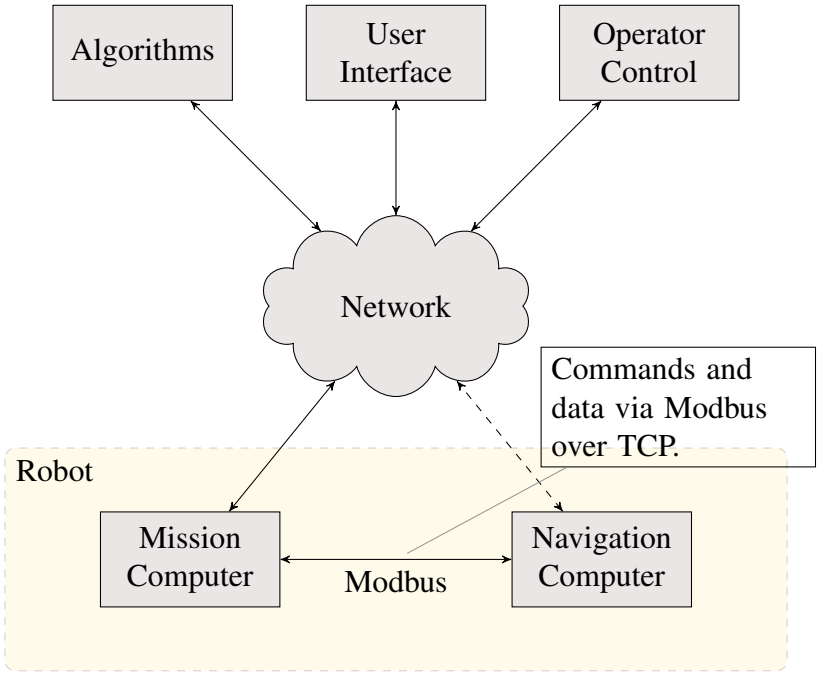


Figure 9.5: AURES deployment diagram. Each robot has a navigation computer for low-level control over sensors and actuators, and a mission computer for high-level control. These two computers communicate through *Modbus over TCP*.

Sensors

- Two dual-band (L1 and L2) GPS receivers
- DGPS based on GSM communication link
- Main odometry sensors: accelerometers, gyros, magnetometer and rotary encoders

9.2.4 Software

The robot software system is written in cross-platform C++. The software architecture is designed on the principles of a modular architecture with layered functionality. Each robot is deployed in the AURES system as depicted in Figure 9.5. The navigation computer handles sensors and actuators, stabilization and navigation between two given points. The software system described below resides on the robot's mission computer.

A number of executable components exist on each robot; these are depicted in Figure 9.6. Adhering to an extendable, modular architecture, each component has a delimited responsibility and functionality.

Robot Controller Handles communication with the navigation computer. Responsible for mode specific functionality and transitions between the five states described below.

Video Controller Handles a video encoder card and serves as multicast video source.

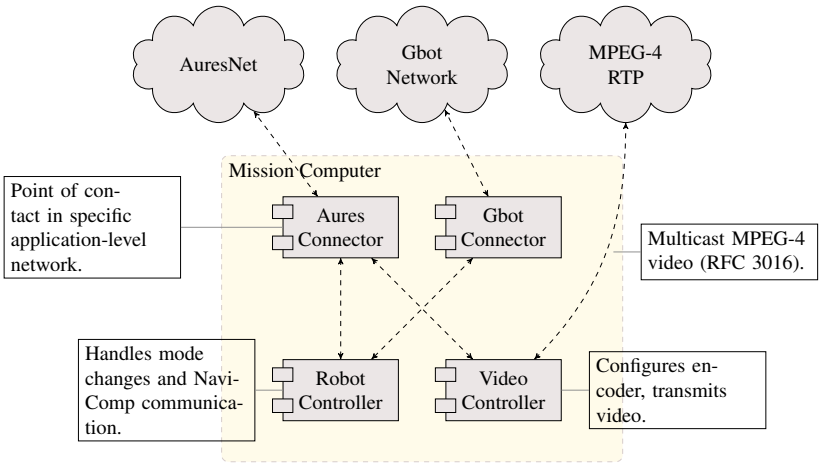


Figure 9.6: GroundBot component diagram. Each executable component has delimited responsibility and functionality.

AURES Connector Point of contact in the application specific AURES network. Routes messages to and from the other components.

Gbot Connector Optionally present as point of contact in Rotundus' GroundBot network including more detailed messages.

9.2.4.1 Robot Controller

A robot can be in any of the five states *Not Ready*, *Error*, *Ready*, *Waypoint Following Mode*, and *Direct Feedback Control*.

At start time, a robot enters the Not Ready state and immediately tries to go into the Ready state. In the Ready state, the robot waits for commands from a controller, such as the AURES Controller or the Operator Control. If it receives a *waypoint following* command, a route plan to follow, it enters the Waypoint Following Mode state. It goes back to the Ready state if it receives a halt command or if it finishes the route plan.

If the robot receives a *direct feedback control* (DFC) command, it enters the Direct Feedback Control state. DFC commands are used to set speed and heading or roll angle and to control the cameras manually. The robot goes back to the Ready state when it receives a halt command.

If an error is detected in any of the states, the robot enters the Error state. It tries to correct simple errors such as navigation computer communication failures automatically. Otherwise, it stays in the Error state and waits for operator intervention to solve the problem.

9.2.4.2 Video Controller

Each robot is equipped with an video encoder for each of the two cameras. These encoders produce MPEG-4 video compliant with the International Standard ISO/IEC 14496-2 [1]. The encoded video is multicast in accordance with RFC 3016 [22]. Details of the video's multicast source address and port is mediated through AURESnet.

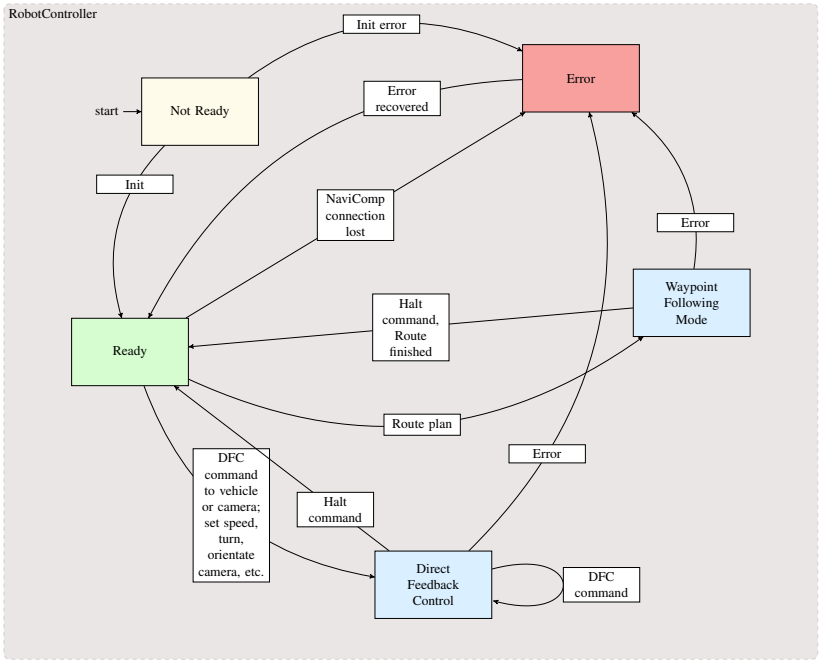


Figure 9.7: GroundBot state chart. Each transition is labeled with the command or event that triggers it.

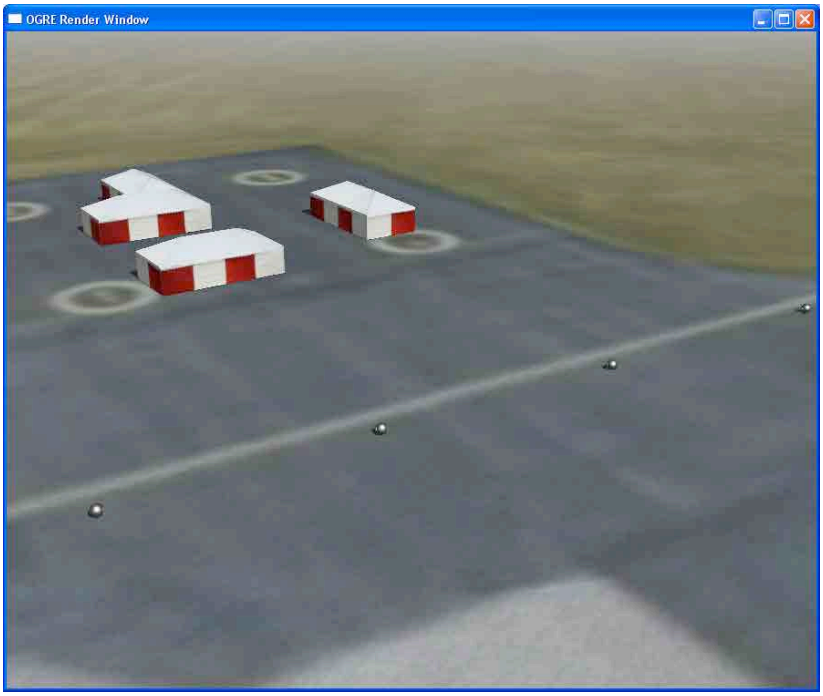


Figure 9.8: The simulator used in the project.

9.3 Simulator

The simulator is based on the simulation engine Game And Simulation System (GASS) and includes physics, network and visualization functionality, as can

be seen in Figure 9.8, as well as in e.g., Figures 10.6, 10.7 and 10.8. It includes realistic UGV models and interfaces to the AURESnet. Since it uses AURESnet, the simulated UGV are treated in exactly the same way as the real UGVs. In fact, the AURES Controller does not know what UGVs are real and what are simulated. The simulator can display images from all UGV cameras, as well as top views from arbitrary camera positions. These video streams can furthermore be broadcast over the network and recorded by e.g. the SSIC.

9.4 AURESnet

AURESnet is an API for system communication. It serves as the common communication interface for all system nodes (components).

AURESnet uses the "RakNet" network API[2] and takes care of the connection of nodes and message handling. It has a multiplatform capability which is necessary since the complete system consists of components running either Linux or Windows. The defined messages are inspired by STANAG 4586, a standard for interoperability between UAV, Unmanned Aerial Vehicles, and ground control stations. Example of defined messages are:

Vehicle Steering Command Allows for direct control of the UGVs. Used by the Operator Control.

Payload Steering Command Allows for direct control of the UGVs' sensors. Used by the Operator Control.

Telemetric data Contains the navigated position and orientation of the UGV and the actual parameters of the sensors, e.g., zoom grade, and azimuth and elevation angles.

Route plan Contains waypoint trajectories and include information about desired time of arrival for each waypoint and sensor directions.

UGV Status Status message that informs for instance about battery level and WLAN signal strength.

9.5 Operator Control

The Operator Control (OP) supplies a graphical user interface (GUI) that allows for manual control (tele-operation) of one UGV at a time.

The OP computer is equipped with a touch screen and the tele-operation GUI, shown in Figure 9.10, contains a number of on-screen buttons for different functionalities. From top to bottom, the buttons have the following names and functions.

Drive Pressing this button puts the GUI in drive mode. Moving the joystick will produce vehicle steering commands.

Search Puts the GUI in search mode. Moving the joystick will produce payload steering commands. The GUI can not be in both drive and search mode at the same time.

Zoom In Causes the UGV's cameras to zoom in.

Zoom Out Causes the UGV's cameras to zoom out. Manually, the cameras' zoom angle can be controlled in ten steps from maximum to minimum field of view.

[2] http://www.jenkinssoftware.com/

(a) The OP computer.

(b) The OP in use.

Figure 9.9: The touch-screen Hammerhead tablet used as OP computer.

U-Turn Pushing this button switches what is considered front and back of the UGV for manual control. Essentially, a 180° turn on the spot.

UGV Choice Pushing this button present the user with a list of available UGVs to choose from.

Off This button exits the GUI after positive confirmation in a dialogue message box.

Reconnect Closes and re-opens the connection to the AURES network.

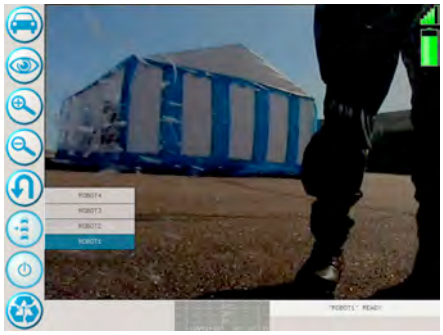
In Figure 9.10, the operator has selected *Robot 1*. By putting the GUI in Drive mode, the selected UGV has been put in Direct Feedback Control mode. Apart from buttons, the main part of the GUI is devoted to the video from the selected UGV. The operator can point in the video image as a fine-grained control of the payload. The video is overlaid with battery status and network signal strength of the UGV. A status text pertaining to the UGV is presented in the lower right corner. Straight below the video image is a widget that shows the current payload attitude with a miniature of the video image. The operator can point in this widget for coarse-grained control of the payload attitude.

When the GUI is started, no UGV is selected. By pressing the UGV Choice button, the operator is presented with a list of present UGVs. This can be seen in Figure 9.11(a). By pressing one of the available UGV options present in the list, the GUI switches to show the video and other data from the selected UGV. This is shown in Figure 9.11(b). Since no commands have yet been sent, the UGV *Robot 1* is in the Ready state.

Just like the UGV software system, the OP GUI is written in cross-platform C++. Implemented as a Model–View–Controller variant [18], separation of

Figure 9.10: Screenshot of the Operator Control GUI. The operator has selected *Robot 1* and put it in Direct Feedback Control Mode.

(a) No robot chosen.



(b) Robot 1 chosen.

Figure 9.11: Screenshot of the Operator Control GUI showing the UGV choice list, before and after a particular UGV has been selected.

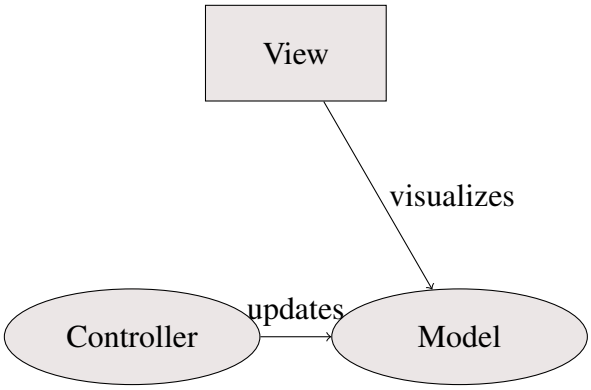
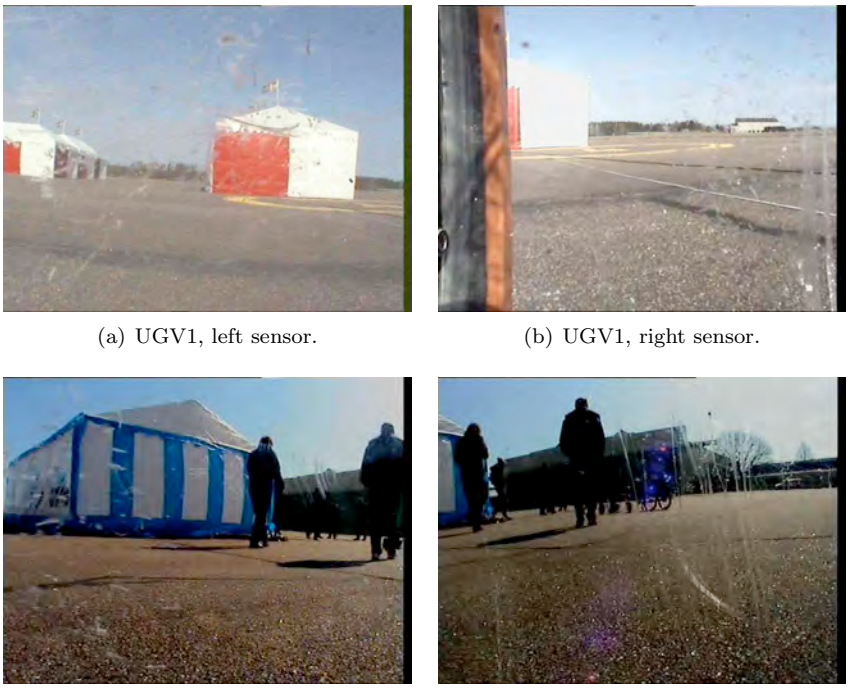


Figure 9.12: The Model–View–Controller paradigm separates data, updating of the data, and visualization of the data.

models (for example, UGVs and world data) and views makes the GUI very flexible. The GUI as described above is an adaptation, with a specific view, of a GUI used in research about using 3D visualization for UGV tele-operation [38].

9.6 User Interface



(a) UGV1, left sensor.

(b) UGV1, right sensor.

(c) UGV2, left sensor.

(d) UGV2, right sensor.

Figure 9.13: Single video frames from the left and right sensors of two UGVs.

The User Interface has a number of functions:

- Gives the operational picture

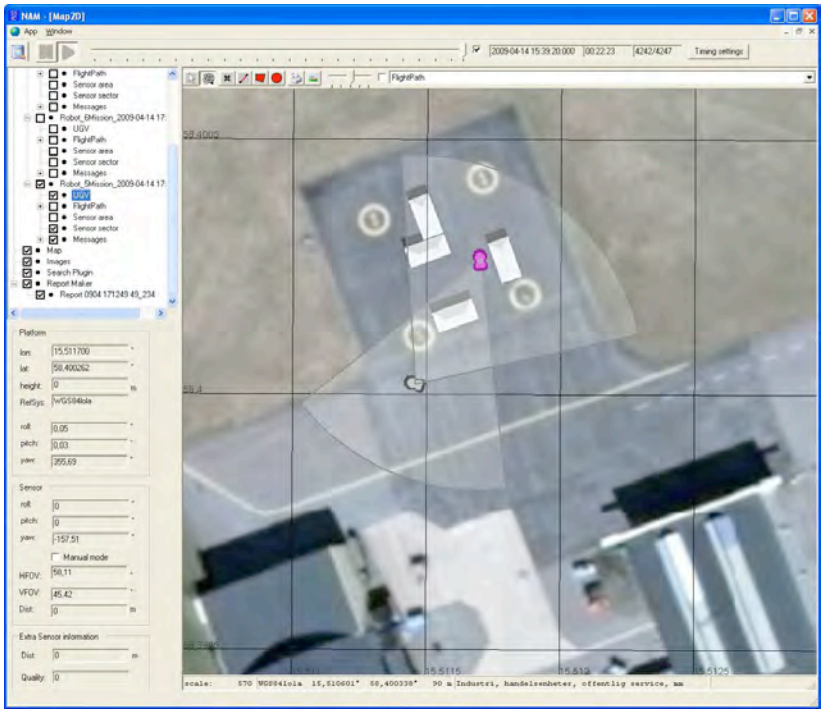


Figure 9.14: SSIC, Sensor Source Intelligence Cell, constitutes parts of the User Interface. SSIC is a system for storage and evaluation of sensor data developed for the Swedish UAV system *Ugglan*. In AURES it is used to give the operational picture and store UGV and sensor information.

1. Displays the connected UGVs (both physical and simulated) in a map.
2. Displays the sensor coverage by the respective UGVs.
3. Displays the video streams from the onboard sensors

- Preserves data to allow for replay of missions
  1. Record all telemetric information
  2. Record video streams
- Controls the selection and initialization of the algorithms.

The User Interface gives the operational picture and constitutes a command and control system for the cooperating UGVs. Figure 9.13 shows example of video information from the UGV sensors and Figure 9.14 illustrates SSIC, Sensor Source Intelligence Cell. SSIC displays the UGVs and sensor information in a map and record data during operation. It also includes functions to evaluate images and create intelligence reports.

9.7 AURES Controller

The AURES Controller, shown in Figure 9.15, is a combined algorithm graphical user interface (GUI) and scenario editor. The bare minimum for running the algorithms is the AURES Controller together with the simulator, shown in Figure 9.8. As seen in Figure 9.15, the main part of the GUI shows the

scenario, with buildings and outer boundary designated by blue spheres, and the current UGV positions designated by white arrow. Note that the arrows correspond to the positions of the UGVs in the simulator screenshot, Figure 9.8. These positions are updated through the AURESnet communications. The lower right part of the GUI shows the buttons invoking the different algorithms, while the lower middle part shows what UGVs are present on the network, and thus ready to receive commands. The left parts of the GUI are used for scenario editing, i.e., placing buildings and boundaries.

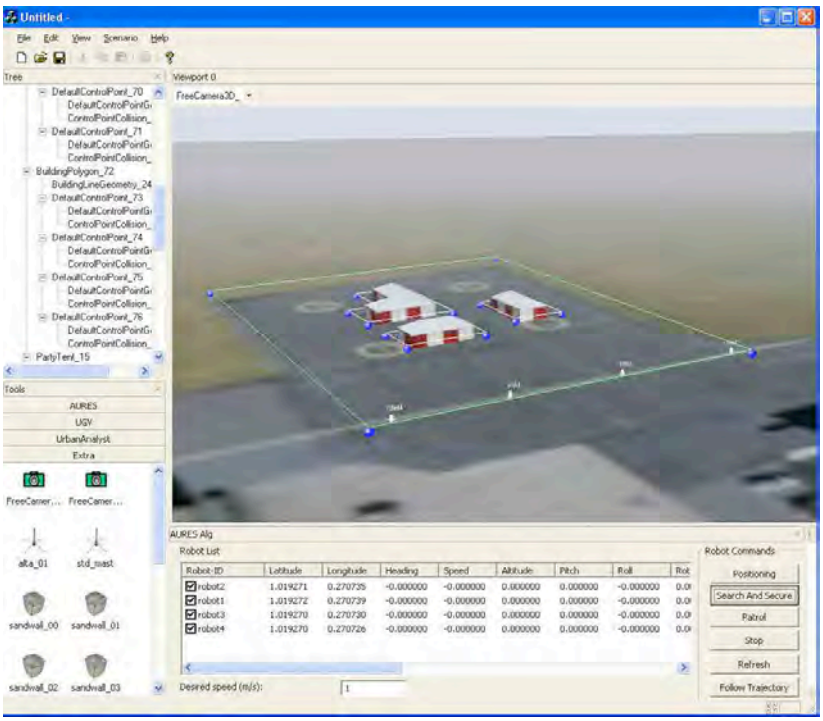


Figure 9.15: The AURES Controller, a combined algorithm GUI and scenario editor.

10 The Demo

This chapter contains a brief description of the demo event that took place in Linköping on the 15 of April, 2009. During the day the customers were first given a presentation of all the results of the project, roughly corresponding to the contents of this report. We then went outside to look at the Groundbot UGVs and the group of tents making out the demo area. In the command and control tent, the AuresController application with the user GUI was shown wirelessly connecting to both simulated and real UGVs in the mission area. Screens showing real camera video streams where sitting next to monitors showing what the simulated UGVs saw. Unfortunately, electromagnetic disturbances degrading the performance of both the wlan and the on-board compasses made the UGVs unable to track the way-point paths computed by the algorithms, so interactive runs with the simulated robots, see Figures 10.6, 10.7, 10.8, 10.9 and 10.10, together with video recordings of a few successful real UGV runs from the previous day were used, see Figure 10.1,10.2 and 10.3. After that the customers tried controlling the Groundbots themselves using the hand-controller.

Apart from the customers from FMV, the demo was also attended by a TV team from Vetenskapsmagasinet, on SVT. They were filming the Groundbots for most of the day, but also took the time to interview the program manager from FMV and the FOI project manager, see Figure 10.4.

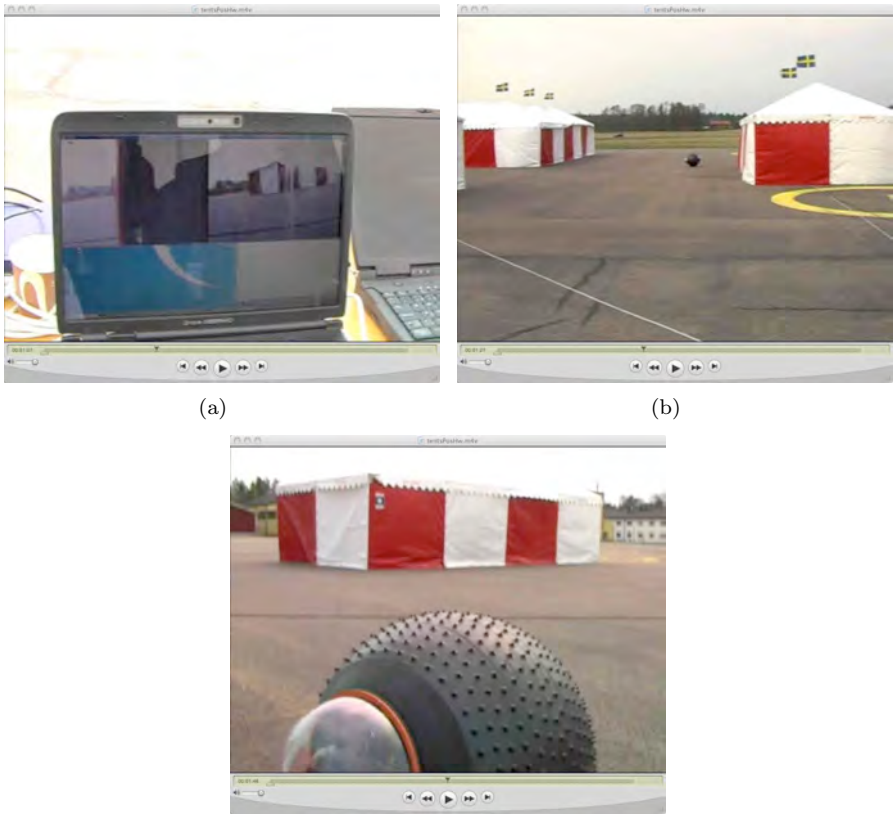
Figure 10.1: A Groundbot running a positioning scenario has stopped to cover two walls of the building ahead.

(a) (b)

(c) (d)

(e) (f)

Figure 10.2: A set of snapshots from a movie recording of a mixed hardware/simulation wall coverage mission. The real UGV covers the north and east wall of the closest building while the simulated UGV covers the south wall. The snapshots continue in Figure 10.3



(a) (b)

(c)

Figure 10.3: A continuation of the movie snapshots in Figure 10.2. The final snapshot is the same as in Figure 10.1.

(a) (b)

(c) (d)

(e) (f)

Figure 10.4: Six photos from the demo event. In (c) the FMV program manager is interviewed by SVT.

Figure 10.5: The command tent hosted all the computers of the complete Aures system depicted in Figure 9.1.

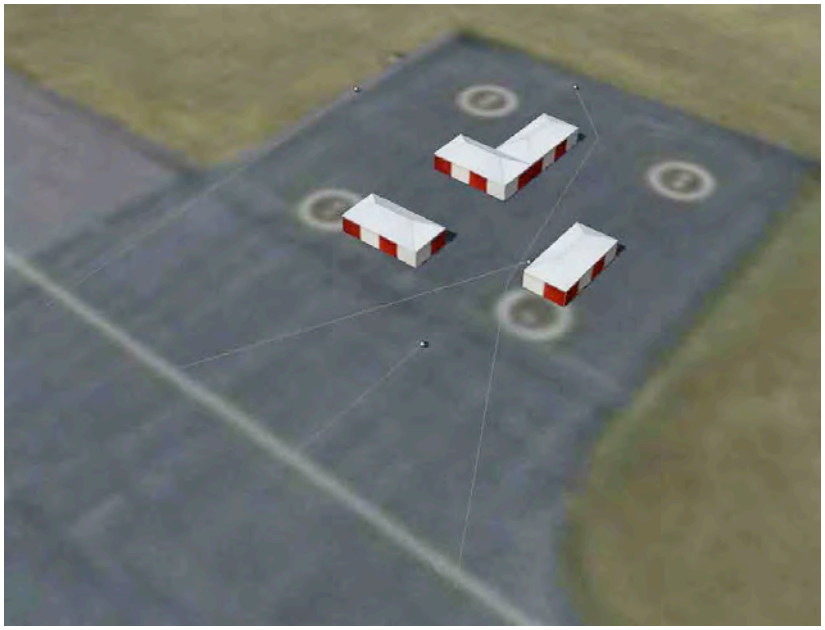


Figure 10.6: A simulated positioning scenario that was run at the demo site. The UGVs are to cover the two leftmost buildings.

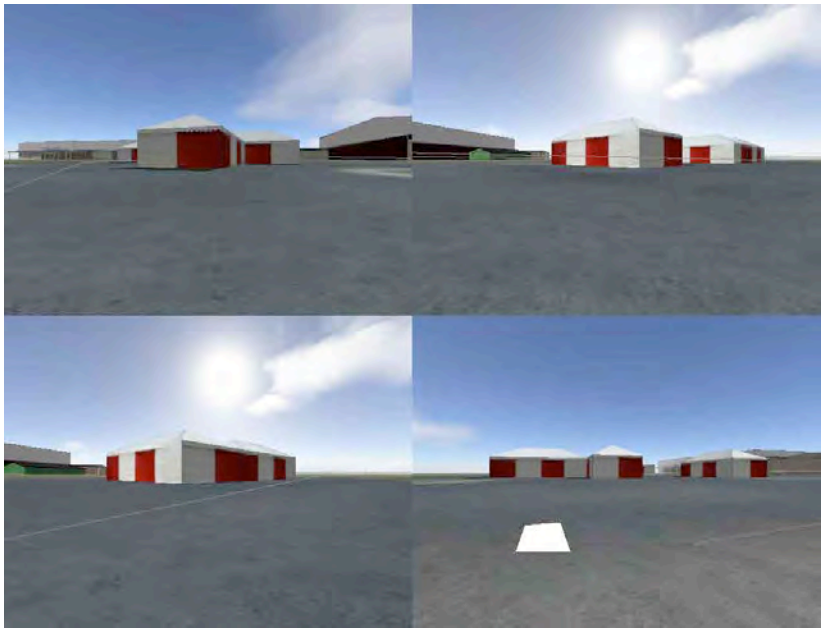


Figure 10.7: The UGV cams of the scenario in Figure 10.6 above.

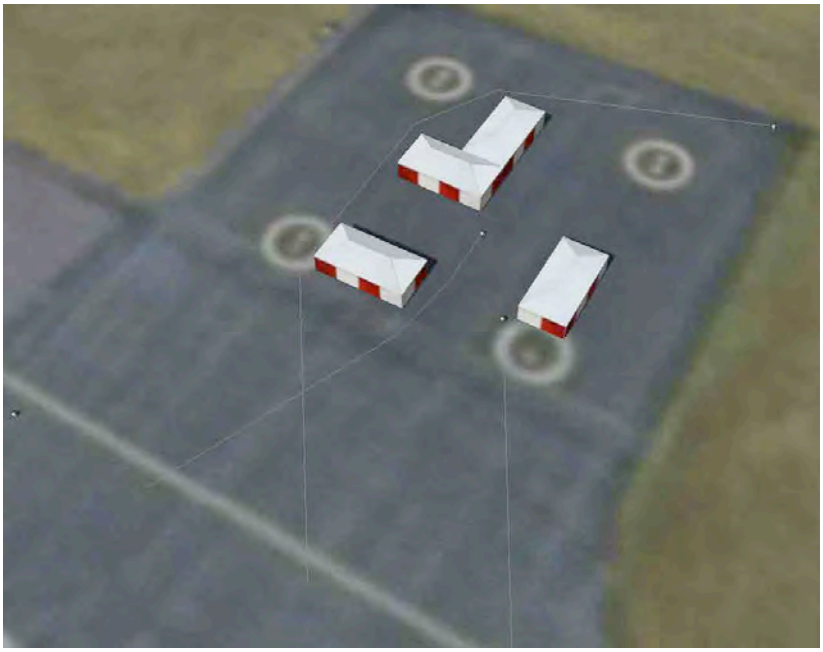


Figure 10.8: A simulated patrolling scenario that was run at the demo site. The UGVs are to cover all free space between and around the buidlings.

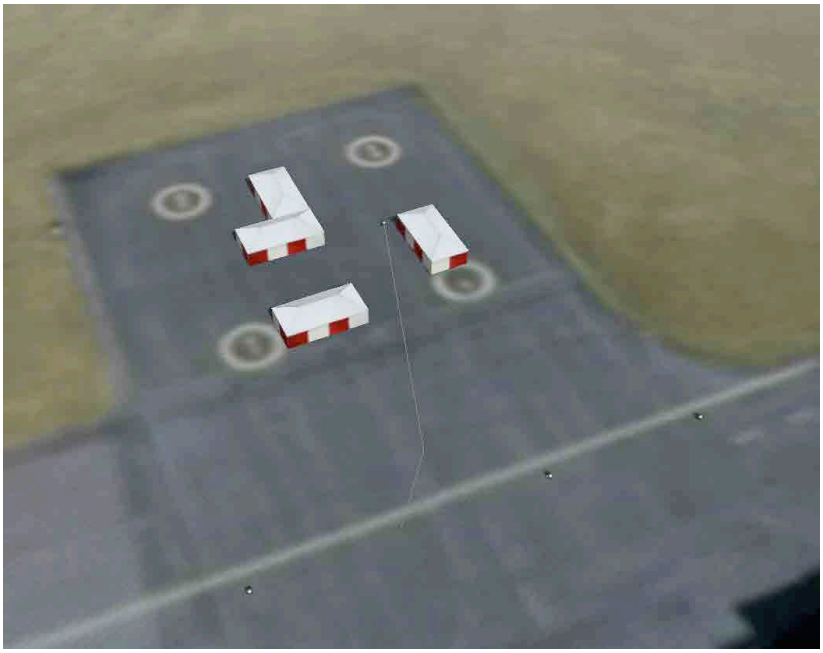


Figure 10.9: The first stage of a simulated search and secure scenario that was run at the demo site. The blocker is positioned to enable the searcher in Figure 10.10 below to secure the area.

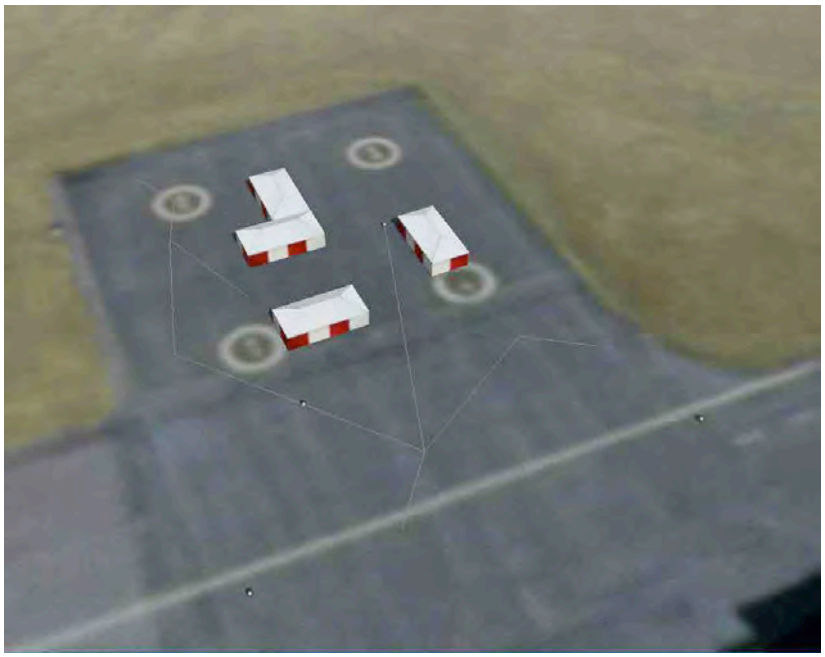


Figure 10.10: The continuation of the scenario in Figure 10.9 above. The searcher has visited the rightmost branch of the path and continues up the leftmost branch to complete the search and thus secure the entire area.

Figure 10.11: The AURES Team, including the program managers from FMV.

11 User Requirements and Evaluations

This chapter contains an overview of the results from the user interviews that was carried out both at the beginning, and at the end of the project, [40, 41]. The interviews were carried out with representatives from the security company Securitas, firemen at Södertörns Brandförsvar and the Swedish Armed Forces in terms of F17, P4, K3, FHS, MSS, Swedec and the Camp Northern Lights in Afghanistan. In the first interviews, a number of application areas were a UGV-system could be useful were identified. The civilian applications included surveillance of factories, power plants, airports and harbor areas as well as teleoperated UGVs doing search and rescue, or acting as a bomb-robot, doing Improvised Explosive Device Demolition (IEDD). The military applications listed were reconnaissance and surveillance of either military bases, weapon depots, or international camps. These results were used in creating the scenario described in Chapter 1, and are described in somewhat more detail below. In the second interviews, short movie clips illustrating the different algorithms were shown and discussed. The results of these interviews are described at the end of this chapter.

For civilian surveillance the possible applications included, as above, factories, power plants, airports and harbor areas. The motivation for introducing UGVs in these areas is cost reductions, and not so much risk reduction, which is more important in the military applications. A necessary constraint in these application is that the UGV operator can be located at a central facility, with the possibility of controlling many UGVs at different locations, rather than being in the vicinity of the UGVs. The areas where a flexible UGV system can compete with stationary cameras and other sensors are furthermore assessed to be those where valuable goods are stored temporarily, such as airports or harbor areas. Having personnel mounting and removing static cameras all the time is not cost effective compared to an adaptive UGV system patrolling the constantly changing area. The trend in the civilian security business is believed to continue towards automation and the fraction of tasks where high tech equipment is used was estimated to grow from 8% to 30% in the near future.

There are three main military surveillance applications: military bases, weapon depots, and international camps. At military army or air bases, there is a need to patrol the outer perimeter to make sure no one enters the area without permission. The trend in this area is towards fewer, but more qualified and better equipped personnel, due to the fact that the number of conscripts in Sweden decreases rapidly at the same time as patrol dogs are becoming less available. As in the civilian case, UGV patrols are considered to be particularly useful in situations where a lot of valuable equipment is stored temporarily somewhere, which is often the case before and during big exercises.

Weapon and supply depot surveillance is an important application in Sweden due to the fact that there are many such depots spread throughout the country as a result of strategic defense planing. If an alarm is received from such a depot, a manned patrol sets out to check it. If armed criminals are trying to access the depot, such missions can be very dangerous. Therefore, the patrol proceeds with care when approaching the depot and establishing control of the surrounding area. In such situations, patrolling, line-of-sight perimeter and wall coverage are natural tactics, as well as searching and securing the interior of a depot. The UGV cameras are also believed to be useful in collecting evidence from an intrusion, as well as for establishing and breaking contract

with armed criminals in a controlled way.

Possible surveillance objects during international missions are camps, airbases and, as above, temporarily stored equipment. An UGV-system is believed to be useful in a number of situations. Examples include the initial situation, when the camp has no established outer barrier or fence line, as well as during *Battle Group* missions, which are too short for establishing such barriers. UGVs can furthermore be used to patrol separation lines between two opposing forces, and in areas that are not suitable for manned patrolling, such as places where there might be mines, or un-detonated munitions. Finally, UGVs can be used to temporarily secure an area in cases of arrests, important meetings, forward headquarters or local celebrations, such as weddings.

For military reconnaissance a UGV-system could be useful in military operations in urban terrain (MOUT), both when entering buildings and when advancing along streets. When entering buildings a situation close to the weapons depot case described above occurs, where many of the group capabilities described in the scenario of Chapter 1 are useful. When advancing in urban terrain UGVs could be used for both situational awareness and, if equipped with the appropriate sensors, detection of improvised explosive devices (IEDs).

Teleoperation of UGVs is extremely important in surveillance, IED-demolision as well as search and secure applications. A common, and well documented fact, see Chapter 4, is that the operators would like to be able to complete these missions faster. This would for instance increase the probability of rescuing people from burning houses, and decrease the probability of bombs detonating or the stationary UGV operator being shot at by hostile snipers.

Having reviewed the findings of the first set of user interviews we now describe the results of the second set, where movie clips of the algorithms were shown and discussed. The overall views were that the developed algorithms were useful for partially solving the problems. To increase the usability of the UGV system a number of adjustments and extensions were requested, such as

- Adapt patrolling not to disturb other activities in the area.
- Combine patrolling and wall coverage.
- Perform iterative non predictive patrolling.
- Adapt patrolling to the case that the operator takes direct control of one UGV
- Adapt search and secure to indoor environments such as weapons depots.
- Combine advancement, patrolling and line-of-sight perimeter surveillance.
- Adapt surveillance behavior to remain concealed from a possible intruder.

The potential users also wanted to be able to try the algorithms in an even more realistic setting, corresponding to the nature of their missions.

All the requests above are natural next steps to carry out as future research if funding so permits.

Bibliography

[1] *ISO/IEC 14496-2 - Information technology – Coding of audio-visual objects – Part 2: Visual*.

[2] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3), 2005.

[3] I. A. Akyildiz, T. Melodia, and K. R. Chowdury. Wireless multimedia sensor networks: a survey. *IEEE Wireless Communications*, 14(6), 2007.

[4] D. Anisi, T. Lindskog, and P. Ögren. Algorithms for the connectivity constrained unmanned ground vehicle surveillance problem. In *IEEE European Control Conference (ECC)*, 2009.

[5] D. Anisi and P. Ögren. Minimum time multi-UGV surveillance. In M.J. Hirsch, C.W. Commander, P.M. Pardalos, and R. Murphey, editors, *Lecture notes in Control and Information Sciences: Optimization and Cooperative Control Strategies*, pages 31–45. Springer, 2008.

[6] D. Anisi, P. Ögren, and X. Hu. Communication constrained multi-UGV surveillance. In *International Federation of Automatic Control (IFAC) World Congress, Korea*, 2008.

[7] D. Anisi, P. Ögren, and X. Hu. Cooperative surveillance missions with multiple UGVs. In *IEEE Conference on Decision and Control (CDC)*, 2008.

[8] D. Anisi and J. Thunberg. Survey of patrolling algorithms for surveillance UGVs. Technical Report FOI-R–2266—SE, FOI, 2007.

[9] David A. Anisi and Petter Ögren. Minimum time multi-UGV surveillance. In M.J. Hirsch, C.W. Commander, P.M. Pardalos, and R. Murphey, editors, *Optimization and Cooperative Control Strategies*, Lecture Notes in Control and Information Sciences. Springer Verlag, 2008.

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

[11] J. Burke, RR Murphy, M. Coovert, and D. Riddle. Moonlight in Miami: An ethnographic study of human-robot interaction in USAR. *Human-Computer Interaction, special issue on Human-Robot Interaction*, 19:1–2, 2004.

[12] Peter A. Buxbaum. Robot Wars. *Defense Technology International*, March/April 2006.

[13] D. V. Dimarogonas and K.H. Johansson. On the stability of distance-based multi-robot formations. In *IEEE CDC*, 2008.

[14] D. V. Dimarogonas and K.H. Johansson. Event-triggered cooperative control. In *ECC*, 2009.

[15] D. V. Dimarogonas and K.H. Johansson. Further results on the stability of distance-based multi-robot formations. In *IEEE ACC*, 2009.

[16] D.V. Dimarogonas and K.J. Kyriakopoulos. On the rendezvous problem for multiple nonholonomic agents. *IEEE Transactions on Automatic Control*, 52(5):916–922, 2007.

[17] A. Fallahi and E. Hossain. QoS provisioning in wireless video sensor networks: a dynamic power management framework. *IEEE Wireless Communications*, 14(6), 2007.

[18] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Resusable Object-Oriented Software*. Addison-Wesley Professional, 1995.

[19] William L. Hays. *Statistics*. Holt, Rinehart and Winston, Inc., 1994.

[20] W. C. Jakes, editor. *Microwave Mobile Communications*. IEEE Press, 1974.

[21] Fotios Katsilieris. Search and secure using mobile robots. Master's thesis, Electrical Engineering, Royal Institute of Technology (KTH), Stockholm, Sweden, 2009.

[22] Y. Kikuchi, T. Nomura, S. Fukunaga, Y. Matsui, and H. Kimata. *Request for Comments: 3016 - RTP Payload Format for MPEG-4 Audio/Visual Streams*, 2000.

[23] M. Lindhé. *On Communication and Flocking in Multi-Robot Systems*. Royal Institute of Technology, 2007. Licentiate thesis.

[24] M. Lindhé. Survey of search-and-secure algorithms for surveillance UGVs. Technical Report FOI-R–2267–SE, FOI, 2007.

[25] M. Lindhé and K. H. Johansson. Communication-aware trajectory tracking. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, Pasadena, USA, 2008.

[26] M. Lindhé, K. H. Johansson, and A. Bicchi. An experimental study of exploiting multipath fading for robot communications. In *Proceedings of Robotics: Science and Systems*, 2007.

[27] M. Lindhé and K.H. Johansson. Using robot mobility to exploit multipath fading. *IEEE Wireless Communications*, 16(1), February 2009.

[28] C. Lundberg. *Assessment and evaluation of Man-portable robots for High-risk professions in urban settings*. PhD thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, 2007.

[29] R.R. Murphy. Human–Robot Interaction in Rescue Robotics. *IEEE Transactions on systems, Man, and Cybernetics, Part C: Application and Reviews*, 34(2), 2004.

[30] R. M. Murray. Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, 129(5), 2007.

[31] U. Nilsson and P. Ögren. Survey of positioning algorithms for surveillance UGVs. Technical Report FOI-R–2268–SE, FOI, 2007.

[32] U. Nilsson, P. Ögren, and J. Thunberg. Optimal Positioning of Surveillance UGVs. *IEEE Conference on Intelligent Robots and Systems (IROS)*, 2008.

[33] U. Nilsson, P. Ögren, and J. Thunberg. Towards optimal positioning of surveillance ugvs. In M.J. Hirsch, C.W. Commander, P.M. Pardalos, and R. Murphey, editors, *Lecture notes in Control and Information Sciences: Optimization and Cooperative Control Strategies*, pages 221–233. Springer, 2008.

[34] P. Ögren. Improved predictability of reactive robot control using control lyapunov functions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[35] P. Ögren. Method for teleoperating an unmanned ground vehicle with a pan camera and such a ground vehicle. In *Swedish patent nr: 0600352-9*. PRV, Patent pending in USA, EU and Israel., 2008.

[36] P. Ögren and P. Svenmarck. A new control mode for teleoperated differential drive UGVs. In *IEEE Conference on Decision and Control, New Orleans, LA*, 2007.

[37] A. Saffiotti, D. Driankov, and T. Duckett. A System for Vision Based Human-Robot Interaction. *IEEE Int. Workshop on Safety, Security, and Rescue Robotics (SSRR-04), Bonn, Germany*, May 2004.

[38] Mattias Seeman, Mathias Broxvall, and Alessandro Saffiotti. Virtual 360° panorama for remote inspection. In *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics*, September 2007.

[39] A. Seuret, D. V. Dimarogonas, and K.H. Johansson. Consensus under communication delays. In *IEEE CDC*, 2008.

[40] P. Svenmarck and P. Lif. Användarbehov för bevakning med flexibla autonoma UGV-system. Technical Report Memo 2081, FOI, 2007.

[41] P. Svenmarck and P. Lif. Utvärdering av algorithmer för bevakning med flexibla autonoma UGV-system. Technical Report Memo 2779, FOI, 2009.

[42] J. Thunberg, D. Anisi, and P. Ögren. A comparative study of task assignment and path planning methods for multi-UGV missions. In M.J. Hirsch, C.W. Commander, P.M. Pardalos, and R. Murphey, editors, *Lecture notes in Control and Information Sciences: Optimization and Cooperative Control Strategies*, pages 167–180. Springer, 2008.

[43] Talon robots – the soldier's choice. `http://www.foster-miller.com/lemming.htm` (2009-04-15).

[44] D.D. Woods, J. Tittle, M. Feil, and A. Roesler. Envisioning Human–Robot Coordination in Future Operations. *IEEE Transactions on systems, Man, and Cybernetics, Part C: Application and Reviews*, 34(2), 2004.

[45] H.A. Yanco and J. Drury. Where Am I? Acquiring Situation Awareness Using a Remote Robot Platform. *IEEE Conference on Systems, Man and Cybernetics*, 2004.

FOI, Swedish Defence Research Agency, is a mainly assignment-funded agency under the Ministry of Defence. The core activities are research, method and technology development, as well as studies conducted in the interests of Swedish defence and the safety and security of society. The organisation employs approximately 1000 personnel of whom about 800 are scientists. This makes FOI Sweden's largest research institute. FOI gives its customers access to leading-edge expertise in a large number of fields such as security policy studies, defence and security related analyses, the assessment of various types of threat, systems for control and management of crises, protection against and management of hazardous substances, IT security and the potential offered by new sensors.

FOI
Defence Research Agency
Defence and Security,
Systems and Technology
SE-164 90 Stockholm

Phone: +46 8 555 030 00

Fax: +46 8 555 031 00

www.foi.se