Robust Area Covering using Hybrid Control

by Manuel Mazo Jr.

Supervised by Karl H. Johansson

A Thesis presented for the degree of
Master of Science in Electrical Engineering

Division of Automatic Control
Department of Signals, Sensors and Systems
KTH, Royal Institute of Technology
Sweden
February 2003

Dedicated to

My family, friends and Erasmus community of Stockholm 2002 and 2003.

Recubrimiento robusto de superficies usando control híbrido

Manuel Mazo Jr.

Presentado para el título de Master of Science in Electrical Engineering
Marzo 2003

Resumen en Castellano

Este apartado trata de ser un breve resumen en castellano del trabajo realizado en esta tesis, donde se aborda el desarrollo de algoritmos e implementación práctica de los mismos orientados a conseguir el recubrimiento de superficies, de forma óptima, mediante robots móviles.

0.1 Introducción

El recubrimiento de superficies de modo eficiente con un vehículo móvil es una tarea común en muchas aplicaciones prácticas. Robots desactivadores de minas, misiones de búsqueda y rescate, y vehículos quita nieves son algunas de estas posibles aplicaciones. Algunas implementaciones en productos de consumo de robots de estas características las encontramos en aspiradoras [1] o cortadoras de césped automáticas [2]. Todas estas aplicaciones comparten el deseo de cubrir una cierta superficie con la máxima celeridad posible. Este tipo de problemas podrían interpretarse como una rama del planeamiento de trayectorias, aunque los resultados de este área no son directamente aplicables a nuestro problema concreto de recubrimiento. En la última decada se ha realizado una gran tarea de investigación en el tema concreto del barrido o recubrimiento de regiones, produciendo grandes logros tales como algoritmos de planeamiento de trayectorias que garantizan el completo recubrimiento de una superficie o leyes heurísticas para realizar tales tareas con un grupo de robots cooperativos.

El propósito de este trabajo es presentar una nueva línea de investigación en torno a un problema específico en el recubrimiento de superficies. En este trabajo consideramos el problema de la minimización del número total de giros necesarios para recubrir una cierta región del espacio, véase, [3].

La principal contribución de este trabajo es la introducción de un problema de recubrimiento en el cual el vehículo encargado de tal tarea presenta un modelo

dinámico que incluye una cierto grado de incertidumbre, representado por un error limitado que se añade a los comandos de dirección del robot. Basados en este modelo y la asumpción de que el posicionamiento del robot es sólo posible en las fronteras de la región que se deseea cubrir, cinco estrategias de control son analizadas a través de una gran cantidad de simulaciones de Monte Carlo y experimentos. El problema tal y cual se ha planteado es bastante realista, como por ejemplo, para robots móviles con capacidades sensoriales limitadas y con un sistema de posicionamiento impreciso. Las estrategias de control desarrolladas son evaluadas tomando como parámetro de interés el número total de giros utilizados dependiendo del grado de incertidumbre presente en nuestros comandos de dirección.

Se muestra cómo para valores de la incertidumbre muy grandes, una estrategia de comandos aleatorios es la que da mejores resultados, lo cual parece razonable intuitivamente, dado que en el caso de tener una gran incertidumbre en nuestros comandos de dirección, el estado del sistema no proporciona mucha informacion útil. En el otro extremo, para valores muy pequeños de error, una modificación basada en heurísticos del simple movimiento de barrido alante-atrás proporciona resultados más que satisfactorios. El caso interesante se encuentra para valores del error en la dirección en un rango medio. En esta tesis se presentan tres algoritmos de control más que mejoran los resultados obtenidos con una estrategia aleatoria o la basada en heurísticos.

Las herramientas computacionales utilizadas en el desarrollo de tales estrategias están principalmente basadas en el software de geometría computacional provisto por [4, 5]. No ha sido objeto del estudio de esta tesis la complejidad algorítmica en profundidad de los algoritmos propuestos ni de sus implementaciones, en cualquier caso en el Apéndice A se encuentra un breve estudio de ellos, y un análisis sobre la complejidad de los algoritmos de geometría computacional utilizados puede encontrarse en [6].

Los algoritmos desarrollados no sólo han sido estudiados teóricamente y mediante simulaciones, sino que también se han desarrollado para su experimentación sobre una plataforma robótica real, Khepera II, y haciendo uso de un pequeño sistema de posicionamiento basado en visión por computador.

0.2 Desarrollo del trabajo

En una primera etapa del trabajo, presentada en el capítulo segundo de la presente tésis, se trata de dar rigor matemático a la formulación del problema a resolver y presentar a su vez las herramientas matemáticas básicas para posteriormente definir los algoritmos y estrategias desarrolladas. En dicho capítulo segundo se encuentran por tanto las ecuaciones de la dinámica del robot, eq. 2.1, para lo cual se adopta un modelo simple en el que se desprecian una gran cantidad de errores, en favor de la única señal de error, error de dirección, en la cual se asume la inclusión de múltiples diferentes fuentes de error. También se definen formalmente las nociones de "región nominal que se cubriría", fig. 2.3, "región que podría ser cubierta", fig. 2.3, y "región que será cubierta con certeza", fig. 2.3, cuando se envía un cierto comando de dirección, teniendo en cuenta que este será modificado por un error aditivo, dando lugar a que el robot se mueva en una dirección que no tiene porque ser exactamente

en la que se le envió.

Con estas nociones definidas, en el siguiente capítulo, se describen los diversos algoritmos o estrategias a utilizar en la sección 3.1. También en esta sección queda justificada con diversas figuras, fig. 3.1, fig. 3.5, la naturaleza híbrida de los controladores diseñados. Además en dicha sección de la tesis se incluye una breve descripción de la implementación práctica de dichos algoritmos.

En el capítulo cuarto se presentan la mayor parte de resultados de la tesis. Se presentan diversas comparativas de los resultados obtenidos por los diferentes algoritmos y estrategias en función de ϵ (que como se explica en secciones previas es la cota a el módulo del error sumado a los comandos de dirección). También se intenta analizar en dicho capítulo algunos efectos particulares de las estrategias aplicadas, tales como valores críticos de ϵ para ciertos algoritmos, o el fenómeno de "atrapamiento en la frontera" (border-trapping) presente en casi todos los algoritmos desarrollados.

Finalmente en el quinto capítulo se presenta la plataforma real, fig. 5.1, sobre la que se llevó a cabo el desarrollo práctico de las estrategias y algoritmos desarrollados. Se describe brevemente el robot Khepera II, asi como la plataforma de experimentación con el sistema de visión. También en esta sección se describe la técnica de detección del robot utilizada en el sistema de visión. Y para finalizar la sección se muestra una captura de un experimento con la plataforma real, fig. 5.4, para su comparación con una captura similar de una simulación, fig. 4.1.

0.3 Conclusiones

Motivados por la necesidad de algoritmos de control robustos para el recubrimiento de regiones para modelos de vehículos con errores, se han presentado y analizado algunas posibles estrategias. Se ha mostrado que el número de giros necesarios para cubrir una determinada región aumenta con el incremento de la cota del error ϵ, como cabía esperar. En estos términos los mejores resultados para valores pequeños de ϵ los ofreció la modificacion basada en heurísticos de un camino boustrophedano, dando números de giros muy próximos al óptimo. Otras soluciones presentaron mayor robustez frente al error de dirección, tales como el algoritmo basado en la noción de "covertura asegurada" o el algoritmo voraz "nominal", el cual obtuvo los mejores resultados incluso siendo el único que no hace uso de la información disponible a-priori sobre el error presente en nuestros comandos. Por tanto se puede también concluir que en términos de robustez y de resultados sobre coste o complejidad, el algoritmo voraz "nominal" es el que presenta mejores cualidades.

A través de experimentación en un sistema real se ha podido probar la viabilidad de la aplicación de estos algoritmos en sistemas reales. Pero también la implementación práctica ha revelado la precisión de la localizazión del robot, en las fronteras de la región a cubrir, como la mayor desventaja de estas técnicas. Si observamos el producto comercial "Solar-mower " de Husqvarna [2], una adaptación de dicho robot podría llevarse fácilmente a cabo para usar las técnicas desarrolladas en este trabajo.

Se ha mostrado el sistema de control en bucle cerrado para el cubrimiento de superficies como un autómata híbrido. De este modo es posible tener un modelo

simple que aun asi captura la complejidad del problema. Podría ser interesante aplicar las herramientas existentes de verificación para analizar los así llamados autómatas temporizados [7], a los cuales el problema de recubrimiento de superficies nos ha llevado.

A parte de lo ya aquí mencionado, en las conclusiones presentadas en lengua inglesa al final de la tésis también se pueden encontrar algunas referencias a más posibles extensiones de este trabajo tales como la posible extensión de este trabajo a un entorno cooperativo con múltiples robots, la extensión del estudio y los algoritmos a regiones de formas más generales o el simple estudio más en profundidad de los algoritmos en función de otros parámetros tales como el tamaño de la región a cubrir o la granularidad de ángulos usada en el barrido realizado por los algoritmos voraces.

Robust Area Covering using Hybrid Control

Manuel Mazo Jr.

Submitted for the degree of Master of Science in Electrical Engineering
March 2003

Abstract

Efficient coverage of an area by a vehicle is a common challenge in many applications. Examples include automatic lawn mowers and vacuum cleaning robots. In this work a vehicle with uncertain heading is studied. Five control strategies based on position measurements available only when the vehicle intersects the boundary of the area are compared. It is shown that the performance depends heavily on the maximum heading error. The results are evaluated through extensive Monte Carlo simulations. An experimental implementation on a mobile robot with localization based on computer vision is also presented.

Acknowledgments

First of all my most sincere gratitude to my supervisor Karl Henrik Johansson, "Kalle" for his excellent guidance, the great discussions and for helping me find and reformulate the problem when I was completely lost. I extend my gratitude also to Mikael Johansson for those interesting discussions with Kalle and me at the beginning of all this project. Thanks to all the people at KTH for their help in all the aspects of my stay in Stockholm, especially to the S3 department for the support given to carry out this thesis, and among them especially to Alberto Speranzon. Thanks also to the people at the Universidad de Alcalá, Alcalá de Henares, and ETSIT-UPM, Madrid for all my education and the especially the latter for giving me the opportunity to come to Sweden to finish it.

I have to thank of course my family for their support during these 5 years of university studies, especially on those bad days when one is so irritating and during these last two years for all the financial support that made possible my stay outside home, to you: *muchas gracias, jamás podré pagároslo.* Thanks to the Erasmus project and European Union for their great work in favor of a better education for all Europeans.

And to finish I would like to thank especially all my friends in Spain, those I met at CERN in Switzerland, and of course and with a special feeling all those Erasmus and not Erasmus students and friends I made in Sweden, to you all for taking me out to drink some beers or just to chat a bit when I was stuck in the middle of nowhere and stressed with the problems of the work.

I should mention also the support provided by Alan Murta on his great open source library for polygon clipping which was fundamental to make this thesis possible.

Of course, thanks to all those that I probably forget, because all seem to be here, but perhaps not all that should be.

Contents

List of Figures

Chapter 1

Introduction

Mine detecting robots, search-and-rescue missions, and snow removing vehicles are applications in which it is important to efficiently cover a given area. Recent commercial implementations in consumer products include automatic vacuum cleaners [1] and automatic lawn mowers [2]. What all these systems share in common is the desire of covering a certain area, and if possible, in the shortest time. These kinds of problems can be seen as a child of motion planning problems, although results in this area are not directly applicable to the coverage problem. Much research has been done in the last decade with great accomplishments as path planning algorithms guaranteeing a complete coverage of an area, or heuristics to accomplish this task with a set of robots cooperating.

The purpose of this work is to present new research line around a specific problem on the area covering problem. We consider the problem of minimizing the total number of turns needed to cover a given area, cf., [3].

The main contribution of this paper is to introduce an area coverage problem that has an uncertain and dynamic vehicle model represented by a bounded additive error on the heading commands to our robot. Based on this model and the assumption that position measurements are only available at the boundary of the area to be covered, five control strategies are analyzed through extensive simulations and experiments. The setup is quite realistic, for example, for mobile robots which might suffer from unreliable position readings and limited sensor capacity. The control strategies are evaluated by comparing this number of turns for various system uncertainties. It is shown that for large uncertainties, a randomized strategy is the best one, which seems intuitive since the system state does not reveal much information in that case. For small uncertainties, a heuristic strategy sweeping the area by a simple back-and-forth motion is sufficient. The interesting case, however, is for uncertainties of middle range. We present three robust control algorithms that then outperform the randomized and the heuristic strategies. The computational tools are mainly based on computational geometry software [4, 5]. The complexity of the coverage algorithms is not studied in the paper. In general, one can probably say, however, that the presented solutions do not scale well. A complexity analysis of the algorithms used in the geometrical operations can be found in [6].

1.1 Background

Many works have been published in the last years about the coverage problem, beginning as part of other problems and at the end leading to the creation of a quite active self research field. Several solutions to the area coverage problem are proposed in the literature, see Choset [8] for a recent survey.

According to Choset's survey one set of approaches are based on heuristics or randomization, which could be applied to our actual problem, although nothing can be found about error on the steering. Heuristics based works have been dealing with finding certain policies or behaviors that help in the coverage aim, especially useful when working with several cooperative robots [9,10]. As well, random policies have been implemented and tested in our work with the particularity of assuring, or better said, detecting when this random-based algorithm has guaranteed complete coverage; and this random based algorithm will be base of comparison with the rest of algorithms designed.

Many existing algorithms consider the decomposition problem, in which the main task is to find intelligent ways to decompose a given large irregular area into pieces easily covered by a default coverage path. This problem is referred in literature as "cellular decompositions". A first approach is named by Choset's work as approximate decomposition, where a fine-grid based representation of the free space is basis for the developed algorithms, and thus work with a discrete representation (graph alike) of the world to design different strategies. In this case, a high grade of reliability is supposed on the robot navigation or a good localization in the space to be able to apply such works in a real application.

Semi-approximate algorithms like Hert and Lumesky [11] deal with the whole coverage problem of a region without performing a complete discretization of the space, but they do not take into account the possibility of errors in the commands. These kinds of decompositions works are more in the line with exact-decompositions, where the aim is to divide an area with obstacles in sub-cells without them.

Among exact-decompositions special interest rises the "Boustrophedon decomposition" [12], where, as explained before, the aim is to divide a complex environment into simpler cells of environment where a simple sweeping motion, known as boustrophedon path, can be performed, assuring a perfect coverage of the whole area. Combining then the first step of this coverage solution, it is the decomposition, and using instead the boustrophedon motion, one of our solutions, a perfect area covering can be achieved in complex environments. In the work of Huang et al. [3], a follow-up of Choset's [12], it is argued that a reasonable optimization criterion is the total number of turns needed for a complete coverage. This is based on the natural assumptions that for mobile robots and other vehicles, turns are costly due to the need to decelerate, turn, and accelerate. In this work the optimization criterion is the total number of turns needed to cover all the subregions. This same optimization criterion was adopted in this work.

It seems like actuator and sensor errors have not been considered in the area coverage literature, though they play an important role for the performance in many applications. It is important to notice that most of the works have been done under assumptions of quite good localization possibility through all the area, while the present study will present solutions under more relaxed localization requirements.

It is also remarkable that the proposed closed-loop control system for the area coverage can be modeled as a hybrid automaton. In this way, it is possible to have a low-order model that still can capture the complexity of the problem. The efficiency to use hybrid control in robotics is also illustrated in time-optimal tracking control problems for Dubin's vehicle [13]. Thus, hybrid automaton representation [7] was adopted in part of the work to present the solutions.

But not just area coverage and hybrid systems literature is a reference point. A great inspiration for notational problem has been the research efforts done by people like in [14, 15] on "Reach Sets Theory"; and as in [14], "Dynamic Games Theory" shows up like an interesting inspiration field, as our problem can be seen as a player trying to drive by discrete commands a robot with aim covering a region, while noise is another player that complicates our task. Although it is a dense field to deal with it in the scope of this work.

On the algorithms complexity issue, we are also reminded of the art gallery problem, which is a somehow related coverage problem that has been extensively studied also regarding algorithmic complexity [16]. In the present work little was studied about the solutions computing complexity, although related to implementation's problems a brief study is presented in Appendix A.

1.2 Work organization

The problem we have faced can be found in the second section of this thesis presented in a more formal and mathematical way than what has been presented so far.

A first natural approach is to look at the problem like a local optimization problem, at each turn, instead of trying to optimize globally. Thus several greedy variants were tested, implemented and simulated, as can be seen in the first part of section 3. Also global optimization was treated, trying to avoid the extremely complicated mathematics behind the problem, and an heuristic based solution obtained by pure exploration is introduced in this same chapter.

With the results obtained after the simulations, which are presented in section 4, and all the developed algorithms, the work shifted to a more practical stage where all these results where tested on a real physical platform. Some other minor problems emerged in this stage of the work, like the positioning problem, and with the solutions some interesting ideas. All this will be presented in section 5. Section 6 concludes the work with a summary of results, as well as some possible work follow ups.

The work here presented has also been base for a paper submitted to the Conference on Decision and Control 2003 (CDC 2003) at Hawaii.

Chapter 2

Problem Statement

2.1 Problem Formulation

Consider the problem of covering the set

$$\Omega = [0, L] \times [0, L] \subset \mathbb{R}^2, \quad L > 1$$

by a square vehicle, as illustrated in Figure 2.1. The vehicle covers a unit square, which is positioned with its upper-right corner at coordinate (x, y). For simplicity, we assume that the vehicle starts in the lower-left corner $(x(0), y(0)) = (1, 1)$. At time $t \geq 0$, the vehicle covers the set

$$c(t) = [x(t) - 1, x(t)] \times [y(t) - 1, y(t)].$$

The accumulated covered set is denoted

$$C(t) = \bigcup_{s \in [0,t]} c(s).$$

Assuming constant unit velocity of the vehicle, its dynamics is given by

$$\begin{aligned} \dot{x}(t) &= \cos(\theta(t) + e(t)) \\ \dot{y}(t) &= \sin(\theta(t) + e(t)), \end{aligned} \tag{2.1}$$

possibly after some feedback linearization of the system, where $\theta \in [-\pi, \pi)$ is the controlled heading and e an unknown angular error. The error, which thus affects the actuation of the control, is bounded by a known constant $\epsilon \in [0, \pi)$.

The vehicle localization is constrained, such that the vehicle position is known only at moments when the vehicle hits the boundary of Ω, i.e., for $t > 0$ such that $c(t) \cap \partial\Omega \neq \emptyset$. This can be implemented in practice by marking the boundary in a suitable way; compare current systems used for automatic cleaning robots [1] and lawn movers [2]. The control strategies studied in the paper are limited to piecewise constant controls triggered by the events $c(t) \cap \partial\Omega \neq \emptyset$, which corresponds to moments when the vehicle turns. Also the error e is piecewise constant and should be interpreted as the uncertainty in the actuation of the turning angle. We suppose that the turning events are separated in time and denoted $0 = t_0 < t_1 < \ldots$. The

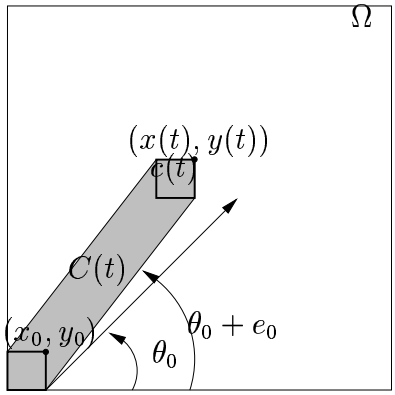


Figure 2.1: Area coverage problem. A square vehicle with uncertain dynamics should cover the area of Ω as fast as possible.

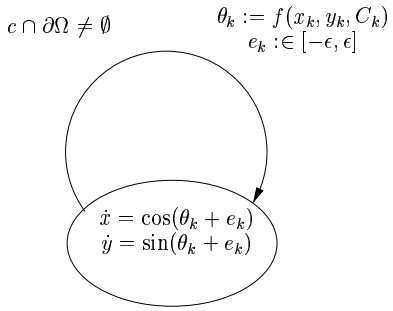


Figure 2.2: A hybrid automaton describing the area coverage control problem. When the guard condition $c \cap \partial\Omega \neq \emptyset$ is enabled (i.e., the vehicle coverage intersects the boundary of Ω), a discrete event takes place. At the event, the control θ_k and error e_k are updated according to a control law f and an error set $[-\epsilon, \epsilon]$, respectively.

control θ at turn k is denoted θ_k and the corresponding error e_k. We suppose that $\theta_k + e_k$ never drives the vehicle outside Ω, i.e., for all $t \geq 0$ we have $c(t) \subset \Omega$.

In order to efficiently cover the area of Ω, denoted $A(\Omega) = \int_\Omega dz$, it is reasonable to try to minimize the total number of turns $N > 1$ made by the vehicle to complete the coverage, cf. [3, 8]. The feedback controls θ_k, $k = 0, 1, \ldots, N$, can be written as

$$\theta_k = f(x_k, y_k, C_k),$$

where $(x_k, y_k) = (x(t_k), y(t_k))$ is the position at the turning point and $C_k = C(t_k)$ is the total covered set up till time t_k. Thus the problem can be rewritten as finding N such that $C_N \supseteq \Omega$.

The closed-loop system can be described as the hybrid automaton in Figure 2.2. When the guard condition $c(t) \cap \partial\Omega \neq \emptyset$ is fulfilled, a discrete-event is generated. It updates the control θ and the error e according to the indicated reset maps. A control law f that solves the coverage problem in N turns corresponds to a family of hybrid trajectories, which each consists of N straight lines.

An interesting hybrid differential game problem is to find a feedback control law

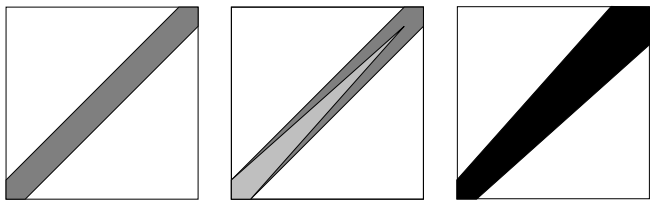
f that minimizes N, given hard constraint on the error $|e_k| < \epsilon$. The authors are not aware of a general solution to this robust control problem. Instead we present a few intuitive algorithms in next chapter, and the rest of the work is devoted to their evaluations.

2.2 Notation

To deal with the problem aim of this work, we will need to define a certain notation and a set of new mathematical tools that can be helpful to compress notation. Here all the already introduced notation will be summarized as well as some new notions used hereon will be introduced.

As in the previous subsection $c(t)$ is the set covered by the robot at time t, i.e. $c(t) = [x(t) - 1, x(t)] \times [y(t) - 1, y(t)]$, and in a similar manner $\bar{c}(z)$ will note the set covered at position $z = (x, y) \in \Omega$. The region to cover will be noted as $\Omega = [0, L] \times [L, 0], L > 0$, its border by $\partial\Omega$ and the interior of the region by Ω^0. Thus having the robot touching the border of the area will be expressed either by $R \cap \partial\Omega \neq \varnothing$, or by defining $z = (x, y)$ as $z \in \partial\Omega$. . Also as already introduced to denote the area of a certain set X it will be used $A(X) = \int_X dz$. The control signal at turn k controlling the heading of the robot will be noted as θ_k and the error present with that control signal as e_k, which will be bounded by ϵ as $|e_k| \leq \epsilon$.

Then we can start defining some basic useful notions for the algorithms description presented in next section. To help define this notation we introduce here $\ell : \mathbb{R}^2 \times [-\pi, \pi) \mapsto \mathbb{R}^2$ as the line $\ell(x, y, \theta) = \{(x + s\cos\theta, y + s\sin\theta) : s \geq 0\}$.



(a) Grey region denotes Nominal coverage

(b) Light grey region is Guaranteed coverage.

(c) Black region is Possible coverage.

Figure 2.3: Different coverages notions. For a $\theta_0 = \pi/2$ and $e_0 = 0.83$[rad]

[Definition: *Nominal Coverage*]Is the subset $B(x, y, \theta) \subset \Omega$, covered from an initial point $(x, y) \in \Omega$ when issuing a control command θ until the robot touches $\partial\Omega$ again. As shown in Figure 2.3.(a) it is mathematically defined as

$$B(x, y, \theta) = \bigcup_{z \in \ell(x,y,\theta) : \bar{c}(z) \subset \Omega} \bar{c}(z)$$

[**Definition:** *Guaranteed Coverage*]Is the subset $B_{\cap}(x, y, \theta) \subset \Omega$ that can be guaranteed to be covered if the bound of the error ϵ is known. Starting from the initial point $(x, y) \in \Omega$ and with control command θ until the robot touches $\partial\Omega$ again. In Figure 2.3.(b) can be seen the shape it takes for a certain example. The mathematical expression for it is

$$B_{\cap}(x, y, \theta) = \bigcap_{\alpha \in [-\epsilon, \epsilon]} B(x, y, \theta + \alpha)$$

[**Definition:** *Possible Coverage*]Defined as the subset $B_{\cup}(x, y, \theta) \subset \Omega$ representing the region that could be covered taking in account all the possible values the error signal can take. Starting at the point $(x, y) \in \Omega$ with control θ it is defined by

$$B_{\cup}(x, y, \theta) = \bigcup_{\alpha \in [-\epsilon, \epsilon]} B(x, y, \theta + \alpha).$$

In a sense this notion is similar to the "reach set" notion as in [14, 15], and takes shapes such as the one represented in Figure 2.3.(c).

Chapter 3

Optimal Area Coverage

Control algorithms to solve the problem presented in the previous section will be introduced here. Such algorithms will act as controllers that issue commands over a moving system plant. The global system can be depicted then, according to the descriptions already given, as an hybrid automaton with a control law f that generates commands based on the positions fed back, see Figure 3.1. In the case of greedy based algorithmic solutions, this f control law has an expression easy to write as will be shown. For the heuristic based algorithm a bit more complex representations will be required.

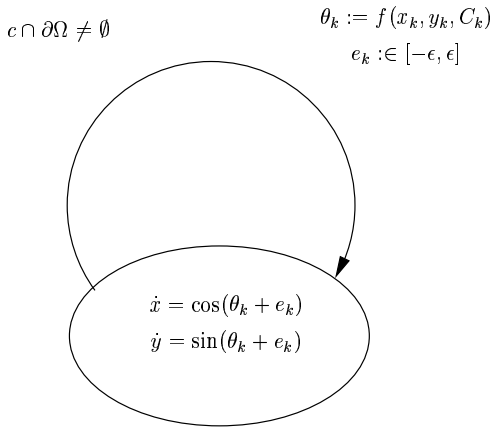


Figure 3.1: System's hybrid automaton description

3.1 Control Algorithms descriptions

Five feedback control strategies for area coverage are presented in this section. They are denoted *nominal*, *guaranteed*, *possible*, *heuristic*, and *randomized*. The first three of them are "greedy" in the sense that they try to maximize the area covered between two turns given different constraints. The fourth strategy is heuristic and basically mimics a traditional way of covering an area when there are no actuator errors; a boustrophedon path [8, 12]. The fifth control strategy is a randomized solution,

which is inspired by commercial implementations in automatic vacuum cleaners and lawn movers [1, 2].

Figure 3.2 shows a comparison of how the first three control strategies are derived. The snapshot is taken at turn $k = 2$. The current coverage $c(t_2)$ of the vehicle is marked by a small square. The gray area corresponds to the accumulated coverage $C(t_2)$. At this moment, the hybrid control strategies maximize the area to be covered till turn $k = 3$, i.e., search for the best control θ_2 over the interval $[-\pi, \pi)$. Figure 3.2 shows areas for $\theta_2 = -\pi/4$.

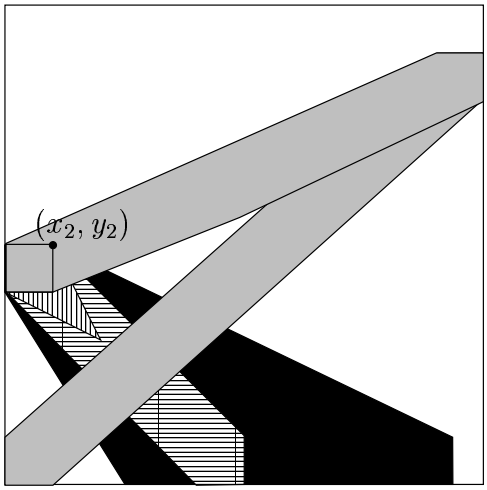


Figure 3.2: Comparison of area coverage control algorithms at $t = t_2$.

3.1.1 Nominal Control

The *nominal* control strategy maximizes the new area covered by the vehicle between turn k and $k+1$ neglecting the influence of the error. The feedback control is given by

$$\theta_k^{\mathrm{N}} = \arg \max_{\theta \in [-\pi, \pi)} A(B(x_k, y_k, \theta) \cup C_k),$$

The union of the striped areas corresponds $B(x_2, y_2, -\pi/4) - C_2$ in Figure 3.2.

3.1.2 Guaranteed Control

The *guaranteed* control strategy maximizes the new area that is guaranteed to be covered by the vehicle between turn k and $k+1$. This feedback control law is given by

$$\theta_k^{\mathrm{G}} = \arg \max_{\theta \in [-\pi, \pi)} A(B_\cap(x_k, y_k, \theta) \cup C_k),$$

In Figure 3.2, the vertically striped area corresponds to $B_\cap(x_2, y_2, -\pi/4) - C_2$. Note that this algorithm will not work for large ϵ. In that case, when a sufficiently large part of the area has been covered at time t_k, say, the guaranteed new area to cover is equal to zero, i.e., $A(B_\cap(x_k, y_k, \theta_k) \cup C_k) = A(C_k)$. (When this happens in our implementation, a random control action is issued.)

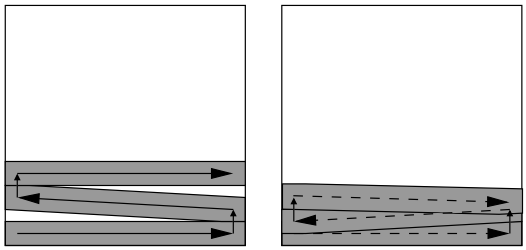
3.1.3 Possible Control

The *possible* control strategy maximizes the area that corresponds to the nominal control but evaluated over the union of all possible errors less than ϵ. This feedback control law is given by

$$\theta_k^{\mathrm{P}} = \arg \max_{\theta \in [-\pi, \pi)} A(B_{\cup}(x_k, y_k, \theta) \cup C_k),$$

The union of the black and the stripped areas in Figure 3.2 corresponds to $B_{\cup}(x_2, y_2, -\pi/4) - C_2$.

3.1.4 Heuristic Control



(a) Pure boustrophedon path control in presence of steering error.

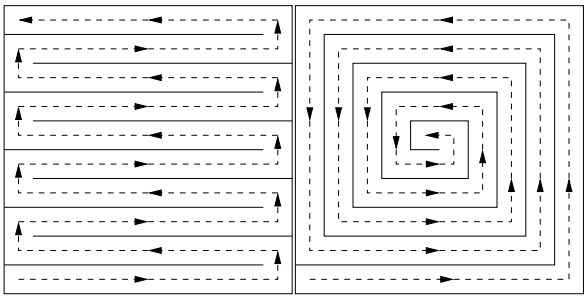
(b) Heuristic algorithm developed behaviour.

Figure 3.3: The left picture shows that a boustrophedon path does not succeed to cover the set Ω, when there is a non-zero steering error ϵ. The proposed heuristic strategy, however, performs conservative movements to guarantee complete coverage, as shown in the right picture.

The *heuristic* control strategy mimics a boustrophedon path, which is the simple back-and-forth motion an ox follows when dragging a plow in a field [8]. The only difference here is that the heuristic control θ_k^{H} is chosen conservatively, so that $C(t)$ is guaranteed to be a connected set for all $t \geq 0$, see Figure 3.3. Note that a pure boustrophedon strategy, without the error compensation, might not succeed in covering the whole set Ω.

The reasoning followed to design such algorithm starts finding trajectories to follow in case there is no error on the commands. An optimal trajectory to cover any area is the known as "Boustrophedon path" which basically consists in back and forth parallel movements as can be seen in Figure 3.4. Although this is the trajectory we will use for developing the heuristic algorithm, it is not the only one that gives

a minimum number of turns to cover a certain area, but also a "rectangular-spiral" movement would satisfy coverage in the minimum number of turns. This two paths for a unit length robot cover the given square (and rectangular if approached in the right direction) in a optimal $\hat{N} = 2L - 2$ number of turns.



(a) Boustrophedon path (b) Rectangular spiral path

Figure 3.4: No error optimal trajectories

The basic idea behind this algorithm is trying to follow as closely as possible one optimal trajectory motion assuring that the error will not modify the robot trajectory in a way that will remain uncovered pieces of the area if the same motion structure is followed. It is, in the case of using a boustrophedon path as base to the algorithm, we will try to cover parallel straight bands while sweeping from bottom to up side of the region. This strategy will be translated in long movements as parallel as possible to the bottom border interleaved with short robot-sized movements toward the top of the region following a border when the long side-to-side sweepings reach a border. This can be described with an hybrid automaton as in Figure 3.5.(a).

Hence if some error is present in our direction commands the resulting trajectory could leave spaces without being covered as shown in Figure 3.3

Thus to assure this lost of area to cover will not take place, an obvious solution is forcing the robot motion in its side-to-side sweeps to go in a direction pointing up as most as to be parallel to the band covered in the previous side-to-side sweep. To assure such condition means that no space will be left uncovered when moving toward the top border, in between two side-to-side movements.

Knowing the maximum values the error can take this condition is easily fulfilled modifying the control slightly to take in account this error maximals ϵ. In someway this modification can be seen as shifting the error values from $[-\epsilon, \epsilon] \rightarrow [0, 2\epsilon]$ or $[-\epsilon, \epsilon] \rightarrow [-2\epsilon, 0]$. But also instead issuing always as commands $\theta = -\epsilon$ or $\theta = \pi + \epsilon$ that would be the modified commands of $\theta = 0$ and $\theta = \pi$, should be noticed that after each side-to-side movement, a pseudo-redefinition of horizontal movement is done, taking as reference of horizontal the last side-to-side trajectory followed (including the error in the realization). Thus, as can be seen in the automaton, the commands take as expression: $\theta_k = -\epsilon + ((\theta_{k-1} + e_{k-1}) - \pi)$ in the side-to-side movements to the right, and $\theta_k = (\pi - (\theta_{k-1} + e_{k-1})) + \epsilon$ when to the left. All this

is depicted in the automaton description in Figure 3.5.(b).

In the practical implementation used in our simulations some slight modifications have been taken over the automaton description, those are:

1 Starting movement from top-right corner following border till lower-left corner. Which will make easier a practical implementation where the covered area is traced after each movement, to detect the moment of total coverage.

2 It is assumed that in the movement along the lower border no error is present due to the ability of the system of trajectory correction as on the borders we assume localization, which give us such correction capabilities.

3 If, due to the additive error, the robot hits the lower border, it will move along it in the same direction it was following until it hits a vertical border.

3.1.5 Randomized Control

The *randomized* control strategy is simply to let θ_k^{R} take a random value from the uniform distribution $\mathcal{U}(-\pi, \pi)$. This algorithm is easy to implement, since no state information, such as current position or covered area, is needed. The Electrolux automatic cleaning robot Trilobite [1] and the Husqvarna automatic lawn mover Solar Mover [2] apply similar randomized navigation schemes.

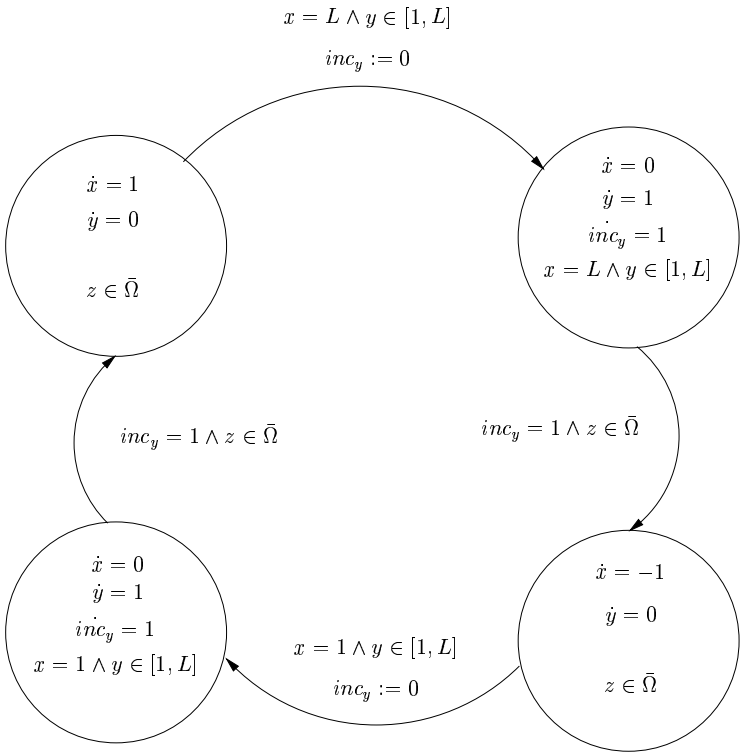
3.2 Computational Implementation

It should be noticed that our greedy algorithms defined as they are in the previous section try to maximize the area covered after a certain turn, that it turns to be the same as minimizing the area remaining uncovered. Hence, as it is explained in Appendix A, the practical implementations for speed and simplicity reasons are realized not using this notion of area covered but the complementary idea of uncovered area remaining.

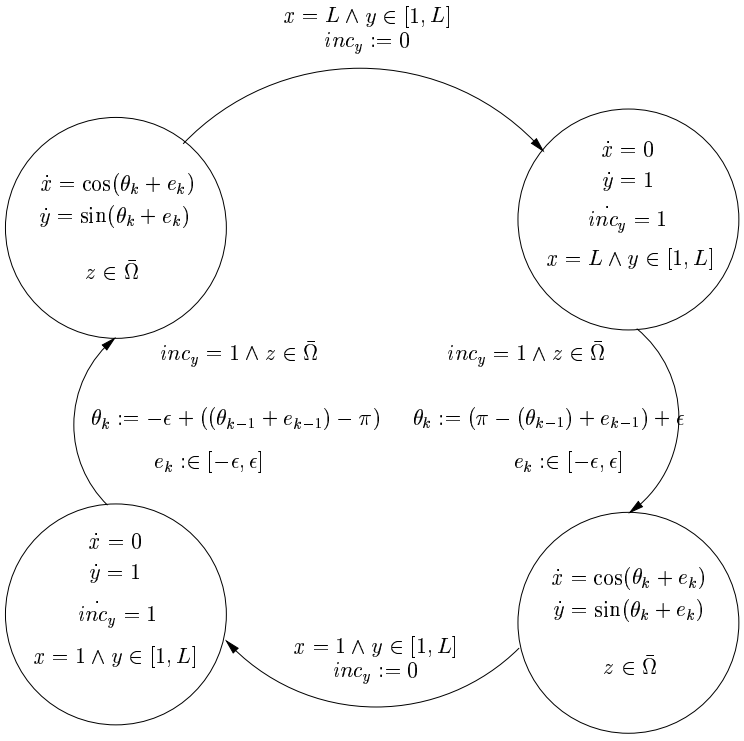
Also in the mathematical description of the algorithms, maximization over a continuous space on the angle variable is assumed, while in the practical implementations of the algorithms this maximization is obtained by sweeping a finite set of angles. This issue that obviously would affect the algorithms performance has not been studied, but instead a sufficiently fine discretization was used for the simulations.

Note that although, as we have just said, in a practical implementation we are discretazing the values of θ through the ones we are optimizing, it does not happen the same with the command errors cause once given a direction θ and the bound of the command errors ϵ, the notion of *assured coverage* in such direction is just one and can be calculated without need of any discretization. The same idea applies to *possible coverage* implementation.

The greedy control algorithms require the calculation of the area covered by the polygons generated from the vehicle movements. These geometric algorithms have been implemented in Matlab. They are based on Vatti's algorithms for polygon clipping [4] as implemented by Murta in the General Polygon Clipping Library [5] ported to Matlab through MEX compilation. Functions for area calculation and convex hull generation by Pankratov [17] are also used with slight modifications for Matlab recent versions compatibility.



(a) Boustrophedon Motion automaton



(b) Heuristic algorithm automaton

Figure 3.5: Boustrophedon motion and heuristic algorithm automaton description. Where $\bar{\Omega} = [1, L] \times [1, L]$.

Chapter 4

Analytical and Simulations Results

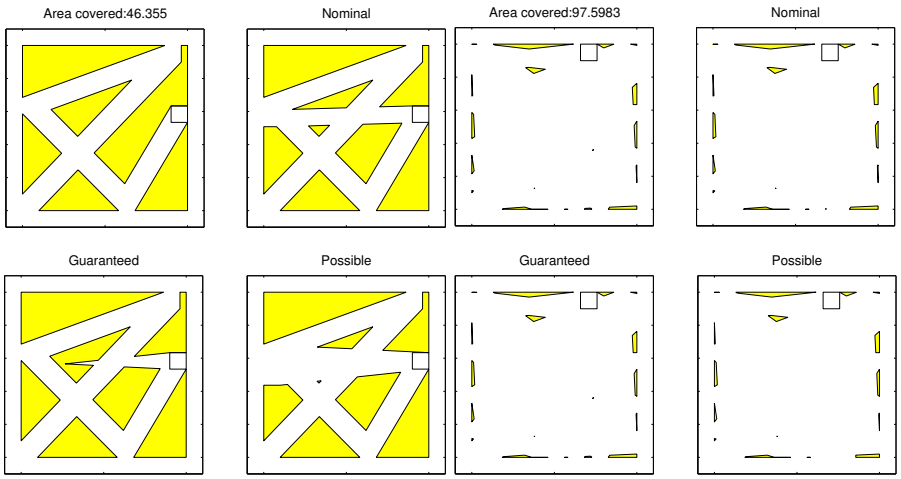
To evaluate the area coverage control strategies, the results from Monte Carlo simulations are presented in this section. The size of the set Ω to be covered is set to $L = 10$. The turning error e_k, $k = 0, \ldots, N$, is drawn from a uniform distribution $\mathcal{U}(-\epsilon, \epsilon)$ (except for the part of the section, where a comparison with normally distributed errors is made).

Figure 4.1.(a) shows a snapshot of a simulation with $\epsilon = 0.078$ at $t = t_4$. The upper left plot shows the accumulated covered set $C(t_4)$ in white, with the current coverage $c(t_4)$ marked by a small square. At this stage the covered area is equal to $A(C(t_4)) \approx 46\,\%$. Recall that the vehicle starts in the lower left corner $(x(0), y(0)) = (1, 1)$. Note that the error e_0 leads to that the vehicle is not able to steer exactly to the upper right corner. The upper right plot indicates the estimation for the nominal control, while the lower left and the lower right shows the guaranteed and possible controls, respectively. Figure 4.1.(b) shows a snapshot of this same simulation at $t = t_{25}$. At this much later state of the simulation almost all of Ω is covered, namely, $A(\Omega) \approx 98\,\%$.

4.1 Performance comparisons

An extensive simulation comparison of the five area cover control strategies is shown in Figure 4.2. For each value of the error bound ϵ marked in the figure, one hundred Monte Carlo simulations were done and the average N was derived for 98 % coverage. The ϵ-axis can roughly be divided into four regions. For small errors ($\epsilon < 0.07$), the heuristic control strategy gives the best result. Note that for $\epsilon = 0$, it gives $N = 18$, which is the optimal. For $\epsilon > 0.07$, the strategy shows quickly bad performance. This is related to the parameter $\epsilon_c = 0.05$, see Section 4.3. For $\epsilon \in (0.07, 0.10)$, the nominal and the guaranteed control strategies are equally good. Then for $\epsilon \in (0.1, 0.4)$, the nominal control is the best. For large errors ($\epsilon > 0.4$), the randomized strategy perform similarly, which is natural because the worst-case error is then larger than 23 degrees. For a given vehicle model, Figure 4.2 indicates hence preferable choices of feedback controls. Though it should be emphasized that the implementation complexity varies for the different control strategies.

It is also interesting to see how influential the error distribution is on the results. Figure 4.3 shows a comparison between errors e_k from the uniform distribution



(a) Late stage when a 46 % of Ω has been covered after 4 turns.

(b) Late stage when a 98 % of Ω has been covered after 25 turns.

Figure 4.1: Snapshots from simulations runs for $L = 10$ and $\epsilon = 0.078$. The current position is remarked with a black square. The figures illustrates an early and late stage of the simulation, showing the covered area, nominal, guaranteed and possible strategies respectively. Note that to take the decisions guaranteed one was used in this particular case.

$\mathcal{U}(-\epsilon, \epsilon)$ (marked with rings) and errors from the normal distribution $\mathcal{N}(0, \epsilon/\sqrt{3})$ (asterisks). The distributions thus have the same means and standard deviations. As expected, the normal distribution yields a slightly lower N, as the for this distribution the probability of errors close to the mean is much bigger; but still the results are comparable.

4.2 Greedy algorithms results

Taking a look to this general comparison picture one could think *possible* algorithm was a waste of time because of its bad results, but still it helps to understand the underlying behavior in these greedy algorithms. While it is pretty obvious the deceitful nature of possible coverage to decide the best direction to take, it sounds quite reasonable to think guaranteed coverage is the best parameter to take in account but still we can see how its results are not as good as one could expect compared to an algorithm where no a priori information is being used.

As we showed in the previous chapter the difference between all this greedy algorithms is in some way the scope of region they are looking at when trying to take a decision. From which could follow that in order the one looking to a thinner scope of region is the guaranteed coverage, followed by the nominal greedy one, and then the one checking a broader region is the possible coverage one. From the curves above it could be concluded that using possible coverage is a too big scope of region that could lead to bad decisions, thus the deceitful nature we were talking about

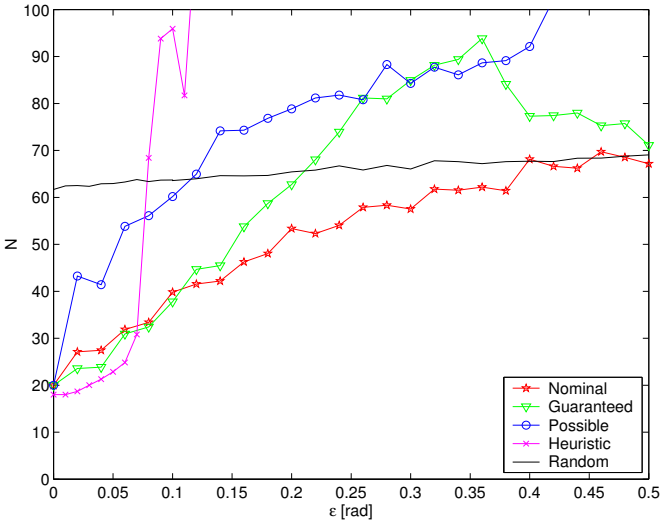


Figure 4.2: The average number of turns N required for 98 % coverage versus error bound ϵ. Five control strategies are compared. Each mark corresponds to one hundred Monte Carlo simulations. The control strategy that gives the best performance depends on ϵ.

before from this algorithm.

The opposite problem arises when thinking of guaranteed coverage, as it is the one checking a smaller region, and the bigger the error, the smaller the region the algorithm takes into account, when the error is big this algorithm looses a general view of the situation, and thus its decisions are conditioned only by a part of the complete picture of the region. In the case simulated, the algorithm checks the surroundings in the borders as was explained in previous sections. Taking a look to the picture in Figure 4.4 one can notice that when $h = (L/2) - 1$ then the central region is never taken in account, and the bigger ϵ is the bigger this region never taken in account is. Then in our case $h = 4$ when $\epsilon \simeq 0.12$ using $h = \frac{1}{2 \tan \epsilon}$.

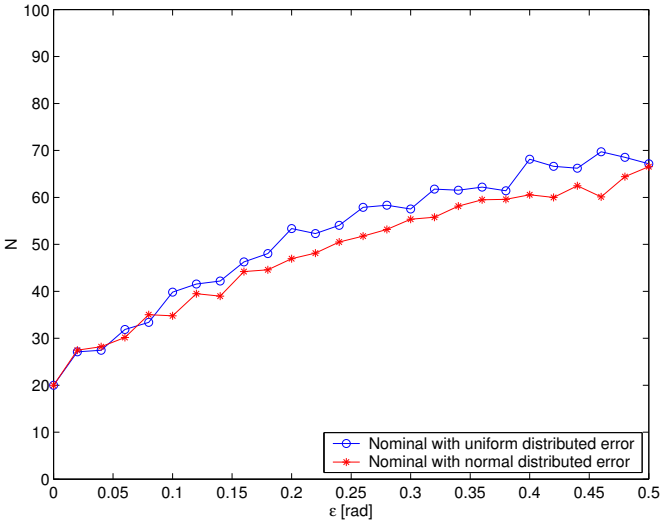


Figure 4.3: Similar simulations as in Figure 4.2 for the nominal control strategy. The rings indicate the results for uniformly distributed errors, while the asterisks indicate normally distributed errors.

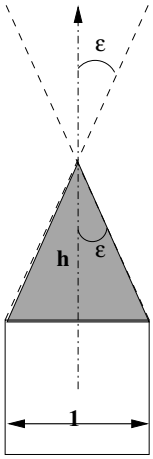


Figure 4.4: Diagram showing the relation between the range covered by *Guaranteed* policy related to ϵ.

Actually, for a certain value of ϵ,it could happen that the region that is checked around the robot is in all directions already covered when Ω is already quite covered. To avoid this null-decision problem, in the practical implementations a random command was issued when this situation arose. If instead this random command were used, the algorithm would stop, then the percentage of area covered would follow the curve in Figure 4.5, where one can see how for a 98% of the area, value that was used to get the previous comparative curves, $\epsilon \simeq 0.12$, and thus from that value on, our algorithm performance is "disturbed" by those random commands that we are forced to issue in the undecidable situations. It turns up also that the point were the assured coverage algorithm starts to be worse than the nominal one, is also

around this 0.12 radians, which makes us think all this phenomena are related.

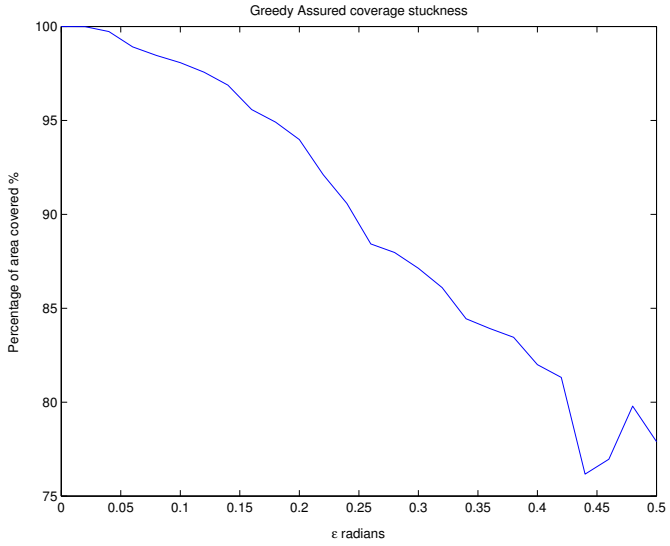
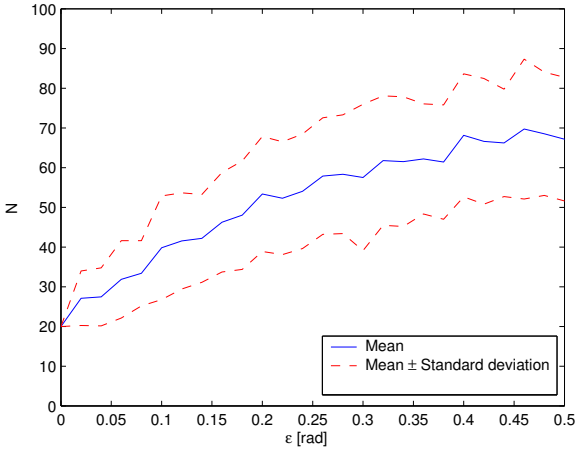
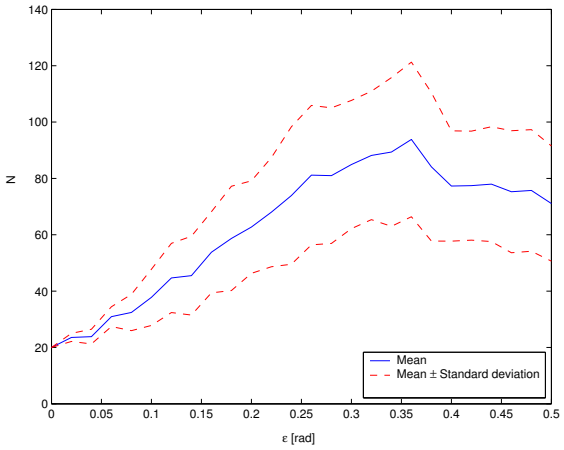


Figure 4.5: Guaranteed greedy algorithm trapping. Percentage of area covered before the algorithm can not take a decision based on the guaranteed coverage parameter.

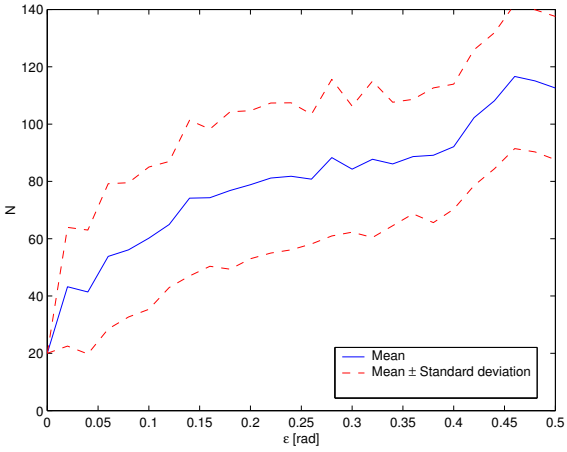
Also can be noticed in Figure 4.6.(b) that the performance of guaranteed algorithm has a peak on the number of turns N. There can be noticed how from that peak on this algorithm tends to give the same number of turns as the randomized algorithm, which is logical, as probably then our implementation of the control is taking too many randomized decisions.



(a) Nominal greedy algorithm.



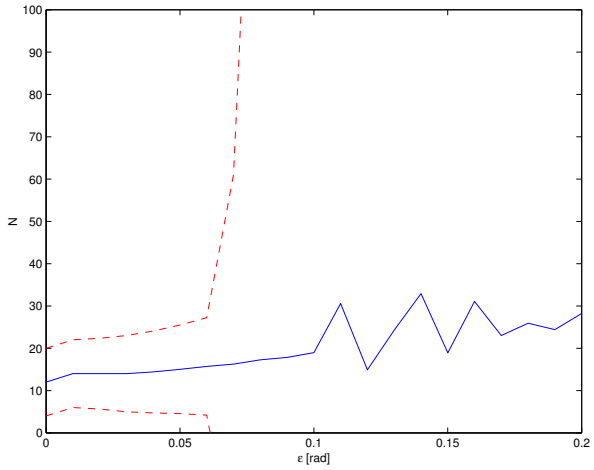
(b) Guaranteed greedy algorithm.



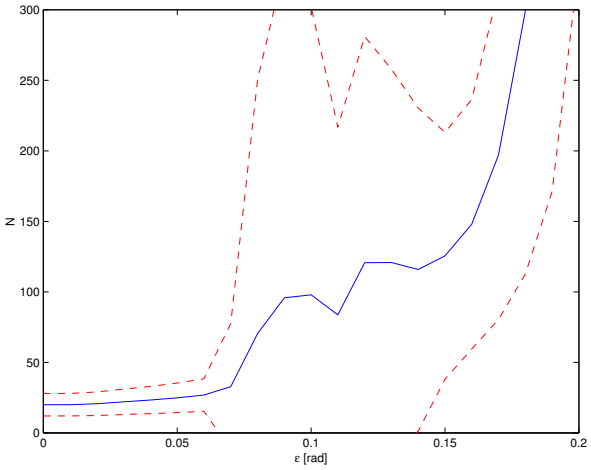
(c) Possible greedy algorithm.

Figure 4.6: Greedy algorithms performance including standard deviation. Obtained from Monte Carlo simulations with 100 runs per point and requiring a 98% of coverage.

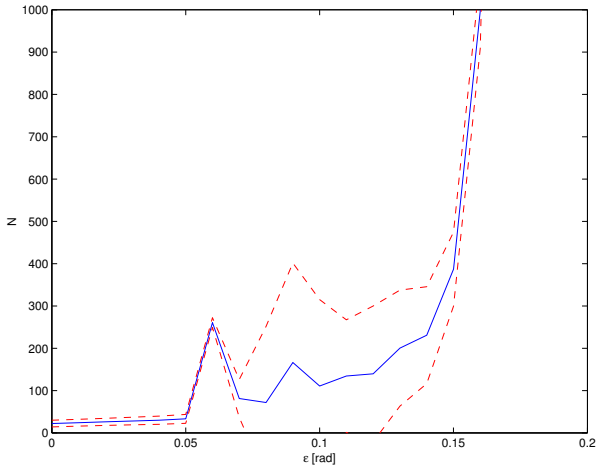
4.3 Heuristic algorithm results



(a) L=7, $\epsilon_{c1} = 0.07$



(b) L=10, $\epsilon_{c1} = 0.05$



(c) L=12, $\epsilon_{c1} = 0.041$

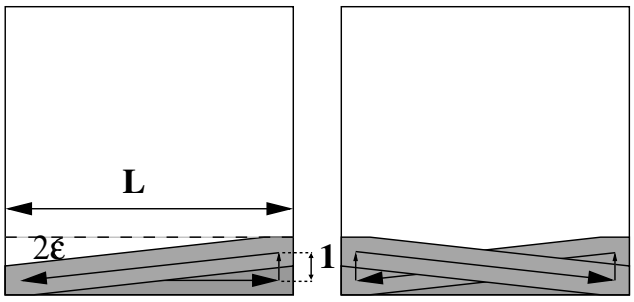
Figure 4.7: Heuristic algorithm performance for different sizes of Ω obtained through Monte Carlo Simulations, 1000 runs per point, for a 98% coverage.

For small ϵ, the heuristic control strategy is efficient in the sense that N is close to optimal. When ϵ grows, however, the strategy rapidly deteriorates. Namely, this deterioration starts at a certain critical value of ϵ for which number of turns required to cover the desired area increases exponentially with ϵ. Thus in the behavior of this algorithm can be found two different regions, a first one where the response of the algorithm is practically linear as showed in the curves in Figure 4.7, and from the critical point over, where response is completely non-linear. It can also be noticed a

second critical point in the algorithm response which divides the non-linear region of its response in a region quite linear and another exponential one.

If looked at carefully one can see how the distance, in ϵ, between the first critical point $\epsilon_{c1}$and the second ϵ_{c2} is approximately the same as the value of ϵ_{c1}, it is $\epsilon_{c2} \simeq 2\epsilon_{c1}$.

To try to explain this result, it can be studied the following situation when applying this algorithm. This situation reflects a possible blocking phenomenon, in which the robot could stay following a closed cyclic trajectory without advancing toward complete the coverage of the aim region.



(a) Critical turn. (b) Cyclic path.

Figure 4.8: Critical point and consequences supporting figure.

As presented in the figures in 4.8, this situation takes place when the error is that big that once applied our heuristic algorithm to pursue the conservative trajectory desired, it could happen that the robot returns to a position in which the advance toward the top-border has been completely compensated and thus it can lead to a cyclic path like the one showed in the Figure 4.8.(b), or even take the robot back from an advanced position toward the lower-border, so that it will have to sweep again an already swept region to reach the coverage area front again.

From Figure 4.8.(a) can be derived for which value of $\epsilon = \epsilon_c$ these situations arises. From simple trigonometry,

$$\arctan \frac{1}{L} = 2\epsilon_c$$

So that in our simulated case where $L = 10$ this is translated in an $\epsilon_c \simeq 0.05$ radians. Which means that for ϵ bigger than 0.05 radians the performance of the algorithm should become worse as this effect would start to take place more frequently and with worse consequences. In the obtained curves, if looking also at the standard deviation, it can be seen how this starts affecting notoriously at a $\epsilon_{c1} \simeq 0.06$, what lead us to the conclusion that this blocking in cyclic paths phenomenon and the noticed critical points are related. The reason for a second critical point would be then, taking into account the position of this one at two times ϵ_{c1}, due to the same situation but now with the worsening that the cyclic effect would appear even without applying our conservative policy that pushes with ϵ radians toward the lower

border.

To prove this theory through simulation, the same kind of Monte Carlo tests were launched but with values of $L = 7$ and $L = 12$, and as can be appreciated in Figure 4.7 (a) and (c) the knee-points of this curves are around values slightly bigger than their respective theoretical ϵ_{c1}, respectively 0.07 and 0.04.

4.4 Border-trapping phenomenon

Another phenomenon worth to mention is the border-trapping (the robot gets stuck in a certain position in the border for several issued commands) present with special importance in all the greedy algorithms and also when running a random policy controller. The problem appears in these kind of controllers where direction commands are just issued in the borders of the region to cover and the robot stops when after issuing one command it reaches again one border. When our robot is placed in the border of the region, based on a decision algorithm of the ones presented, the robot takes a decision on the new direction it should follow. When this direction that has been issued is a command leading to a trajectory closely parallel to the border currently touching, a big or not so big error could push the robots direction toward the border thus making the robot stop right after issuing the command and force the generation of a new command. The probability of appearance of this phenomenon is obviously increasing with bigger ϵ as with a "less parallel" issued trajectory, it can still happen that the error could block the movement against the border.

In a real system or a more complicated simulation this phenomenon could be avoided taking in account our localization capability in the borders. So that in case of having an error pushing our trajectory from an original one nearly parallel to the border to one colliding with that border, a border follow could be performed, and then no turn would be lost.

This phenomenon is also the responsible for a curve like the one presented at Figure 4.9 for a system with a random decision policy. It could be though that as the decision itself is random, adding a random noise would not change the results, and thus it should have a constant response along different values of ϵ, it is, it would require same number of turns to cover Ω regardless of the value of ϵ. But as can be seen in the curve presented, the required number of turns increase with an increasing ϵ, due to the bigger frequency of appearance of the phenomenon here explained.

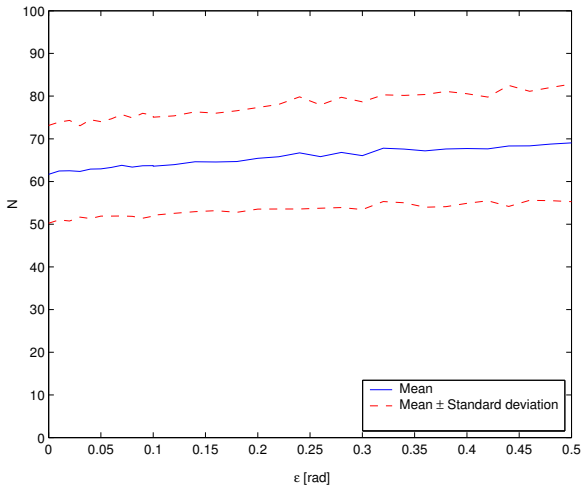


Figure 4.9: Randomized algorithm results obtained from Monte Carlo simulations, 1000 runs per point. Evolution of N depending on ϵ, for a coverage of 98%.

Chapter 5

Experimental Results

5.1 Experimental platform

Once the algorithms to perform the area covering and simulations were developed, these different controllers were tested on a real robotic platform. These were put in practice through the micro-robot Khepera II [18] on a small testing arena of size $L = 550$mm, it is 10 times the diameter of the robot $D = 55$mm, so the experimental setup and the simulation study have roughly the same quota $A(c)/A(\Omega)$. An overhead camera coordinated with the proximity sensors of the robot performed the localization on $\partial\Omega$ task. To reuse the developed code, controlling of the Khepera motion was done also through Matlab, with the speed drawback that it carries. And the localization by computer vision was also implemented in Matlab for easier connection of all the code. We will now go through both two subsystems in the following sections. And then some conclusions from the experimental testings will be presented. A picture of the whole system can be seen in Figure 5.1.

5.1.1 Khepera II Robotic platform

Khepera II is a precision microbot developed by K-Team. Its reduced size (see Figure 5.2.(a)) makes it specially useful for algorithms testing in laboratory environments like those presented in this work. The microbot includes, among others, a set of proximity sensors distributed as shown in Figure 5.2.(b) , from which in our runs just the front set was used to trigger the stop of the robot and launch the camera-based localization algorithm. The other group of sensors interesting for the present work are the two wheels rotation encoders. These are high precision ones that for the present size of Khepera II give a linear displacement resolution of 12 pulses per millimeter. With this data a first theoretical bound ϵ_{th}can be calculated as follows.

Driving one wheel forward and the other backward will make the robot turn around its own a certain angle. The best precision one can achieve is that by means of setting the movement of each wheel to one encoder-pulse in the right direction for each wheel. One pulse means thus a linear movement of 0.083 millimeters on the robot circumference from what can be deduced through simple geometry , $\frac{\alpha D}{2} = 0.083\,mm$, that our minimum turn angle is 0.003 rad. Note that in this bound

(a) Testing arena with overhead camera.

(b) Khepera robot in the testing arena. Note the sizes quota $A(c)/A(\Omega)$.

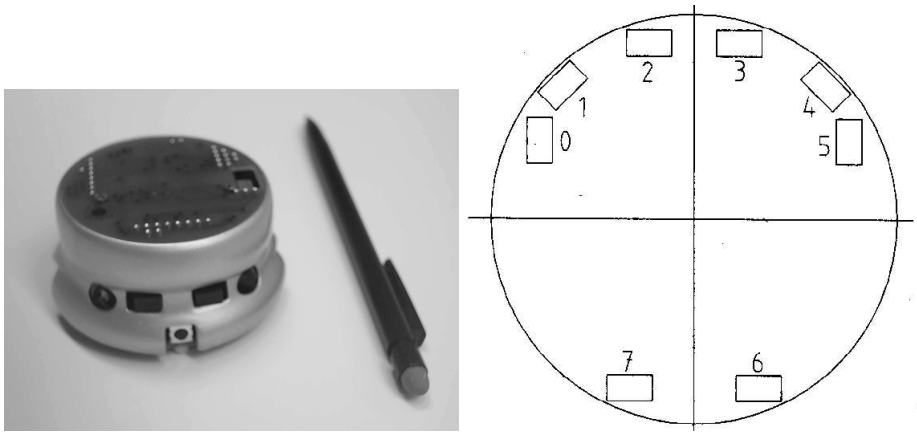
Figure 5.1: Testing arena setup with Khepera II.

many effects are neglected such as robot inertia, that in fact would make this error much bigger. Moreover, in our implementations a reference system is needed for the angular heading of the robot. This reference was obtained by setting the robot heading in a certain known position at the starting point, and in further turns its pointing direction was calculated from the detected position and starting position. Hence more error would be present in the turning control motivated by localization lack of precision.

To obtain a good bound ϵ of the error present in our system, experimentation was used. In this way a more realistic value of the critical parameter ϵ was obtained. The experiments performed were a set of turns with different command angles θ_k, that were compared with the read angles $\bar{\theta_k}$ from the localization system to obtain the turning error $e_k = \left|\theta_k - \bar{\theta_k}\right|$. The system was run for a sequence of turns without reseting the system to a original position to include in this bound also the accumulative effect of the error in the system as is designed. After this experiment, the mean value of the error $\bar{e}$ and its standard deviation σ_e was calculated and the ϵ that would be used for the whole system thereafter set to $\epsilon = \bar{e} + \sigma_e$ obtaining a value of $\epsilon = 0.078$. It should be noted that this value is valid for a certain illumination ambient, as like will be motivated in the next section this parameter played a fundamental role in the localization issue.

5.1.2 Computer Vision localization system

The localization system was achieved by means of an overhead camera connected to the computer and took pictures of the arena triggered by the proximity sensors of



(a) Khepera II micro robot (b) Khepra II IR sensors positions.

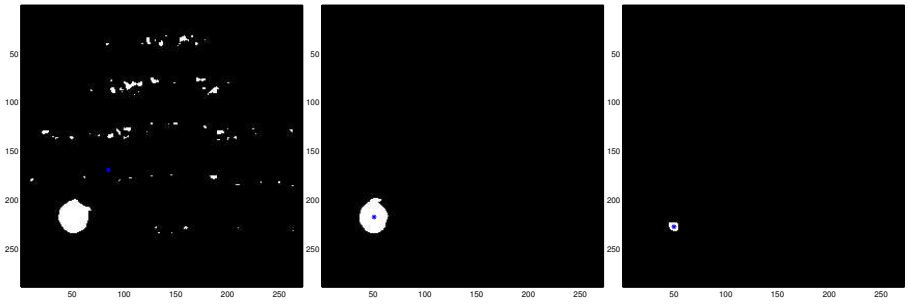
Figure 5.2: Khepera II size comparison and sensors positions.

the robot when they sensed a wall of the border. In fact, an alternative version of the system performed positioning on the borders by taking several pictures through the robot trajectory from border to border and (after treating those pictures) performing a linear regression of all the localized points to calculate the crossing of the trajectory followed with the border. With this latter technique [1] the linearity (as opposed to curved) of the trajectories followed by the robotic system could be also tested, which turned out to be quite good.

The technique used for robot localization in the pictures taken by the camera can be divided in three stages. First the color image was transformed to black and white giving special weights to the different color signals to highlight differences between background and robot, and filter special noise introduced by the camera and illumination. Once the black and white image was available gray level segmentation was performed using a dynamic threshold. This threshold was calculated based on the image histogram, as the gray-level value for which a certain percentage $th_{\%}$ of the image pixels where below. This value $th_{\%}$ is decisive for the performance of the localization and has to be adjusted depending on the illumination, but in general big values around 99% gave results good enough. This 99% value is motivated by the ratio of sizes between robot and area to cover, in our case the robot has a surface of area 1% of the area to cover, hence this 99% as value of thershold. A comparison on the effect of this parameter can be appreciated on Figure 5.3.

With the image segmented into background and robot, the mass center calculation of the points selected as robot was performed, and the resulting value was used as robot position.

[1] Under good illumination environment with the vision system giving precise localization.



(a) $th_{\%} = 98.0\%$ (b) $th_{\%} = 99.0\%$ (c) $th_{\%} = 99.9\%$

Figure 5.3: Grey-level thresholding for image segmentation. Dependence on the $th_{\%}$ parameter. In the images the white region denotes the detected robot, and with an asterisk the detected mass center of the robot is marked.

5.2 Experimental results and conclusions

As illustrated by the snapshot in Figure 5.4, the experiment follows the behavior quite well of the corresponding simulations (Figure 4.1.(a)). When running an experiment for a long time, it has been noticed however that the error model used in the simulation is not accurate. The error distribution tends to change over time. This is particularly the case if the localization error at the boundary of Ω is not negligible, causing in our particular implementation an accumulative effect of the error. And highlighting localization reliance as the main drawback of the proposed control strategies against a purely randomized driving control.

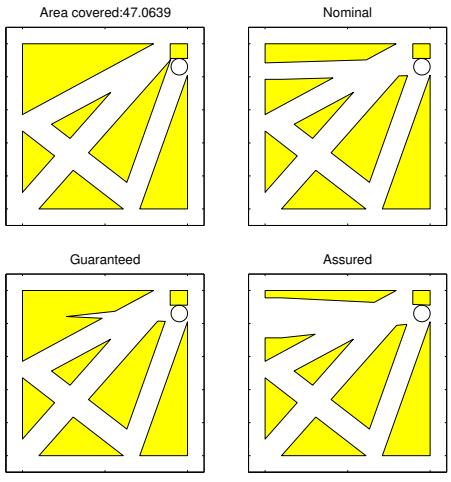


Figure 5.4: Snapshot of a simulation run using $\epsilon = 0.078$ and driving the Khepera II robot with a guaranteed greedy controller.

Still, when illumination was favorable the tested system behaved quite well covering the region in reasonable number of turns N according to the simulated results,

and proving the viability of practical application of the designed strategies.

Chapter 6

Conclusions

Motivated by the need for robust control algorithms for area coverage under uncertain vehicle models, we presented and analyzed a few possible strategies. It was shown that the number of turns needed in order to cover an area is increasing with the error bound ϵ of the turns. Moreover, which algorithm that performed the best depends on ϵ. In these terms the heuristic modification of a boustrophedon path gave the best results given a small value of ϵ, assuring coverage in a relative small number of turns. Other solutions presented better robustness against the steering error, such as the greedy algorithm based on the "guaranteed coverage" notion, and the "nominal" greedy algorithm, which gave the best results even though it does not make any use on previous information on the error present on the system. In robustness sense should be said that "nominal " greedy algorithm was the one showing a better behavior, as well as in a performance over cost or complexity.

Through practical experimentation it has been possible to prove viability of application of such algorithms in real systems. But also practice has revealed localization reliability as the principal drawback of the strategies designed. Taking a look to the commercial product "Solar-mower " available from Husqvarna [2], an adaption of such a system could be possible by discrete magnet marks underground embedded in the same border-detection system. These marks would give a not error-free localization system, where localization error would be again within certain boundaries. In this new setup the mentioned practical problem will arise in the form of a-priori unknown command errors that disturb the system, in contrast with our driving based error, that could be seen as an a-posteriori error, as it appears in the system "after" a new command has been issued.

The closed-loop control system for the area coverage was presented as an hybrid automaton. In this way, it was possible to have a low-order model that still can capture the complexity of the problem. It would be interesting to apply existing verification tools in order to analyze this so called timed automata's [7], which the area coverage problem in this work led to.

Also some more general conclusions could be pursued through Monte Carlo simulations, such as analysis of performance of these strategies depending on how fine is the quantization of steering angles available; on the size L of the region where they are applied; or robustness and performance dependence on the shape of the region to cover. As except heuristic that has more limitations on the kind of shapes it can be used, all the rest of algorithms are applicable to a broad kind of shapes. In this

particular subject boustrophedon cellular decomposition [12] gives an interesting starting point to develop algorithms applicable to more complex region shapes.

Another possible follow-up of this work is the application of these kind of algorithms when a set of robots work cooperatively on the task of coverage, and how they should be modified. For this new problem a possible strategy to follow for this algorithms application is the combination with heuristics rules developed for such cooperative coverage work as in [9, 10]. Particularly reactive, repulsive, behavior making the robots go in opposite directions when a collision occurs will help to spread them in the region and thus be more effective our strategies. But still many problems will still need solution such as localization when those collisions take place, inter-robot effective information share to develop good situation maps and optimize the coverage with each turn or just the development of a more complex set of heuristic rules that would optimize the global system performance.

Bibliography

[1] Electrolux, "Trilobite." http://www.electrolux.com/trilobite.

[2] H. AB, "Husqvarna solar mower," April 1998. http://www.solarmower.com.

[3] Y. Huang, Z. Cao, and E. Hall, "Region filling operations for mobile robot using computer graphics," in *Proceedings of the IEEE Conference on Robotics ans Automation*, pp. 1607–1614, 1986.

[4] B. R. Vatti, "A generic solution to polygon clipping," *Communications of the ACM*, vol. 35, pp. 56–63, July 1992.

[5] A. Murta, "A general polygon library," Jan 2003. http://www.cs.man.ac.uk/aig/staff/alan/software/.

[6] G. Greiner and K. Hormann, "Efficient clipping of arbitrary polygons," *ACM Transactions on Graphics*, vol. 17, pp. 71–83, April 1998.

[7] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[8] H. Choset, "Coverage for robotics - a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, no. 31, pp. 113–126, 2001.

[9] T. Balch and R. C. Arkin, "Communication in reactive multiangent robotic system," *Autonom. Robots*, vol. 1, no. 1, 1995.

[10] D. MacKenzie and T. Balch, "Making a clean sweep: Behaviour based vacuuming," in *AAAI Fall Symposium*, 1996.

[11] S. Hert, S. Tiwari, and V.Lumelsky, "A terrain-covering algorithm for an auv," *Autonom. Robots*, vol. 3, pp. 91–119, 1996.

[12] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon decomposition," in *Proceedings of the International Conference on Field and Service Robotics, Camberra, Australia*, IEEE, December 1997.

[13] A. Balluchi and P. Souères, "Optimal feedback control of Dubins' car tracking circular reference paths," in *Proc. 35th IEEE Conference on Decision and Control*, (Kobe, Japan), pp. 3558–3563, December 1996.

[14] A. Kurzhanski and P. Varaiya, "On reachability under uncertainty," Department of Electrical Engineering and Computer Science, UC Berkeley, Sept 1999.

[15] P. Varaiya, "Reach set computation using optimal control," Department of Electrical Engineering and Computer Science, UC Berkeley, Dec 1998.

[16] H. Gonzalez-Banos and J. Latombe, "A randomized art-gallery algorithm for sensor placement," in *Proc. 17th ACM Symp. on Computational Geometry*, pp. 232–240, ACM, 2001.

[17] K. K. Pankratov, "Saga matlab toolbox," 1995. http://puddle.mit.edu/ glenn/kirill/saga.html.

[18] K-Team, "Khepera ii," 2002. http://www.k-team.com.

Appendix A

Union of a set calculation methods

All the algorithms used in this work have a strong computational geometry charge based on the fact that they have to calculate the union of the sets, polygons in geometrical terms, generated by the robot movements after each turn. This calculation must be done both after one movement has been finish, to update the coverage-map, and before issuing a new command again this kind of operation must be done several times to discover the command that will optimize our maximization problem (just applicable in the case of greedy algorithms). As a critical operation for the performance of our algorithms the study below was fundamental for implementation of the control algorithms.

In the following sections two different approaches to solve this problem will be presented as well as a brief study on the complexity of both approaches to justify the usage of the second approach, indirect method, in the practical implementation.

A.1 Direct method

A.1.1 Theory

A first natural approach to solve this problem is the straight solution, calculating the union of a set of polygons and the area of the resulting polygon. A problem arise when doing so cause the union of several convex polygons in general leads to non-convex polygons, and when we have more than three polygons it also leads in general to polygons with holes. Having non-convex polygons is probably not such a big deal when calculating the area of them, but having polygons with holes could increase the complexity of calculating the area of such polygons too much.

To evade this problem one can try to rewrite the area of the union as a function of areas of intersections. This would simplify the problem a lot, as the intersection of two convex polygons always leads to another convex polygon, and the same is fulfilled for the intersection of several convex polygons (think about the associative law of the intersection).

The solution to this new problem is then given by unfolding the area of the union of all the polygons up to a certain instant of time, into areas of such polygons themselves and areas of intersections of combinations of them. With the support of sets-theory, by induction one can get to the following expression, using $A(p)$ mean-

ing the area of the polygon p. This result is well-known in probability theory by the name of "*Inclusion-Exclusion Formula*".

$$\begin{aligned} A\left(\bigcup_{i=1}^{n} p_i\right) &= \sum_{i=1}^{n} A(p_i) - \sum_{(i_1,i_2):1\le i_1<i_2\le n} A(p_{i_1}\cap p_{i_2}) + \ldots \\ &+ \sum_{(i_1,\ldots,i_r):1\le i_1<i_2<\ldots<i_r\le n} (-1)^{r+1} A(p_{i_1}\cap\cdots\cap p_{i_r}) \\ &+ \ldots + (-1)^{n+1} A(p_1\cap p_2\cap\cdots\cap p_n). \end{aligned}$$

It is important to note that

$$\#\{(i_1,\ldots,i_r) : 1\le i_1<i_2<\ldots<i_r\le n\} = \left(\begin{array}{c} n \\ r \end{array}\right),$$

it is, in each sum we have as many terms as ordered combinations of the n polygons taken in groups of r exist.

Also note that by p_i we refer to the convex polygon generated by the system between the instants $i-1$ and i.

[Proof]:

By induction, for n=1 the result is trivial, then we check for n=2

$$A\left(\bigcup_{i=1}^{2} p_i\right) = A(p_1) + A(p_2) - A(p_1\cap p_2)$$

What is consistent with the geometric theory underlying and a basic result in probability or sets theory.

We have to show now from $n=k-1$ that this formula also holds for $n=k$,

$$
\begin{aligned}
A\left(\bigcup_{i=1}^{k} p_i\right) &= A\left(\left(\bigcup_{i=1}^{k-1} p_i\right) \cup p_k\right) \\
&= A\left(\bigcup_{i=1}^{k-1} p_i\right) + A(p_k) - A\left(\left(\bigcup_{i=1}^{k-1} p_i\right) \cap p_k\right) \\
&= A\left(\bigcup_{i=1}^{k-1} p_i\right) + A(p_k) - A\left(\bigcup_{i=1}^{k-1} (p_i \cap p_k)\right) \\
&= \left(\sum_{i=1}^{k-1} A(p_i) - \sum_{(i_1,i_2):1\le i_1<i_2\le k-1} A(p_{i_1} \cap p_{i_2}) + \ldots\right. \\
&+ \left.(-1)^k A\left(\bigcap_{i=1}^{k-1} p_i\right)\right) + A(p_k) \\
&- \left(\sum_{i=1}^{k-1} A(p_i \cap p_k) - \sum_{(i_1,i_2):1\le i_1<i_2\le k-1} A(p_{i_1} \cap p_{i_2} \cap p_k) + \ldots\right. \\
&+ \left.(-1)^k A\left(\bigcap_{i=1}^{k} p_i\right)\right)
\end{aligned}
$$

In the last expression the elements in the first term are those coming from applying the inclusion-exclusion formula to the polygons $(p_1, ..., p_{k-1})$ while those in the latter term come from applying the inclusion-exclusion formula to the sets $(p_1 \cap p_k, ..., p_{k-1} \cap p_k)$. Then note how the first element contains ordered combinations of cardinality from 1 to n of the polygons in $(p_1, ..., p_k)$ with neither of the entries equal to p_k, while the latter term contains all the ordered combinations that contains one element equal to p_k. Thus, combining both terms taking in account the sign change we prove that the general expression is fulfilled. By induction, it can be concluded that it is true for all n.

A.1.2 Practical implementation and algorithm complexity

The implementation of this method for calculating the covered area is not as trivial as could seem, and several different implementations can be tried. The first and probably worst is to apply directly the expression obtained at the beginning of this section. It is, at a certain moment of time generate all the possible combinations of polygons taken in groups of cardinality from 1 to n, being n the number of polygons up to that moment, and then calculate the intersection of all polygons in such groups. Then just the calculation of the area of such polygons and the accumulation of such values with the proper $+$ or $-$ sign is left.

To get an idea on how complex is the algorithm we can take a look to the number of areas to be calculated when we have the union of n convex polygons. Note that now we are not taking in account the complexity of calculating the different

intersections.

$$\#A(at\ time\ n) = \sum_{i=1}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) = 2^n - 1$$

This expression can be easily proved looking at the Newton's binomial theorem and using $a = b = 1$

$$(a+b)^n = \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) a^{n-k} b^k$$

Some time can be saved if we keep on saving the polygons resulting from the intersections calculated in previous time-instants, and using them to generate the new ones when adding a new polygon to our list of polygons. Still the complexity of the algorithm and the calculus power consuming is increasing exponentially. Applying this rule the number of area computations will follow the same expression, but now at time n we will calculate the number of area calculations as:

$$\#A(at\ time\ n) = \sum_{i=1}^{n-1} \left(\begin{array}{c} n-1 \\ i \end{array} \right) + 1 = 2^{n-1}$$

Because now the combinations are the same as for time $n-1$ adding the new polygon coming at time n and the area of the new polygon alone.

This algorithm can be also implemented on a recursive manner, but in general the number of levels of recursion might be too big, and reusing values calculated in the past may be a complicated task.

A.2 Indirect method

A.2.1 Theory

Another approach to solve the task of calculating the area covered, inspired in the De Morgan law's for Boole-algebra, is to think in the complementary space, it is, think on the area-to-cover. It can be proof that a power set $P(S)$ with $A + B = A \cup B$, $A \,.\, B = A \cap B$, $A^c = S - A$, $0 = \emptyset$, $1 = S$ and $A \leq B \Leftrightarrow A \subseteq B$, defines a Boolean algebra. Thus we can apply the De Morgan law to our problem,

$$A\left(\bigcup_{i=1}^{n} p_i\right) = A\left(S - \bigcap_{i=1}^{n} p_i^c\right) = A(S) - A\left(\bigcap_{i=1}^{n} p_i^c\right)$$

Translating to a more intuitive language, this means that we can calculate the area covered up to a certain moment by, first calculating the polygons generated after each movement that results from the difference of the polygon we want to cover, S, and the recently generated polygon. And then, intersecting all this not-covered-area-polygons generated up to the present, we get the not-covered-area-polygons in the actual instant of time. Thus is just left to calculate the area of this polygons not covered, and subtract it from the whole area to cover to get the area covered.

A.2.2 Practical implementation and algorithm complexity

This new algorithm looks much easier to implement, and as can be easily seen it is far less computing consuming than the first approach. However it can present several disadvantages in a general situation. Such disadvantages arise basically when calculating the complementary polygons. In a general situation the complementary of a certain polygon will be two different polygons and thus the number of polygons to intersect are the double of the polygons one had when calculating the union. Even with this double of polygons, if one compares with the number of polygons to intersect in the first algorithm and the number of areas to be calculated, this new approach shows up to be a much better solution.

It can be shown that with n polygons in general the number of complementary polygons generated after intersections will be as maximum

$$\#P^c(n) = \frac{n(n+1)}{2} + 1$$

and if we do the same under the supposition of a robot generating these polygons as was explained in the introduction, this number can be reduced to

$$\#P^c(n) = \frac{n(n-1)}{2} + 1$$

Again like in the first algorithm, the area covered at time n can be calculated using the results of calculations at time $n-1$. Then we obtain that the number of intersections to be calculated are a maximum of

$$\#\cap P^c(n) = (n-1)(n-2) + 1$$

Thus looking at this results and the disadvantages pointed out for the first approach, it can be concluded that this second approach to the problem is easier to implement and also more efficient in general.