SITUATION AWARENESS AND ADAPTIVE DECISION-MAKING IN AUTONOMOUS SYSTEMS VIA SYMBOLIC LEARNING

A Dissertation in
Mechanical Engineering
by
Xin Jin

© 2012 Xin Jin

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2012
The dissertation of Xin Jin was reviewed and approved* by the following:

Asok Ray  
Distinguished Professor of Mechanical Engineering  
Dissertation Advisor, Chair of Committee

Jeffrey S. Mayer  
Associate Professor of Electrical Engineering

Shashi Phoha  
Professor of Electrical Engineering  
Director, Division of Information Sciences and Technology,  
Applied Research Laboratory

Alok Sinha  
Professor of Mechanical Engineering

Thomas Wettergren  
Senior Technologist, U.S. Naval Undersea Warfare Center Division Newport

Karen A. Thole  
Professor of Mechanical Engineering  
Department Head of Mechanical and Nuclear Engineering

*Signatures are on file in the Graduate School.
Abstract

Autonomous systems are becoming prevalent in the human society as they play a very important role in the applications that need to function in uncertain environments. Such systems usually require integration of several functions including perception and decision-making for operation under incomplete or incorrect a priori information. In this context, situation awareness is the perception of the environment for extraction of pertinent information, and adaptive decision-making is the adaptation to the uncertain environment. This dissertation addresses two main issues: situation awareness and adaptive decision-making in autonomous systems by symbolic learning.

Situation awareness involves the recently developed framework of Symbolic Dynamic Filtering (SDF) and is discussed in the context of supervised learning. In the SDF framework, sensor observations are discretized temporally and spatially to generate blocks of symbols and patterns are generated from these symbol blocks for pattern classification. Research issues such as feature extraction from two-dimensional domain of wavelet-transformed sensor time series, optimization of the symbol generation process, and fusion of heterogeneous sensor information are addressed in this dissertation. The proposed technology is validated by various (experimental and simulated) case studies that include behavior recognition in mobile robots, anomaly detection in nuclear power plants, and personnel detection using multi-modal sensors.

Adaptive decision-making is crucial to ensure effective and efficient operation of autonomous systems in uncertain environments, and is discussed in the context of path planning and autonomous navigation. The search space of the autonomous system is constructed as a symbolic grid map, where the two-dimensional space is partitioned into grids and an alphabet of symbols is assigned to each grid cell to represent the environmental information. Formulation of the search space is then extended in the multi-resolution sense to enable adaptive decision-making based
on the available spatio-temporal information. A navigation algorithm based on the multi-resolution formulation is then proposed for autonomous navigation in uncertain environments. The proposed algorithm is validated on various case studies such as complete coverage of complex and static environments, and simulation of oil spill cleaning in dynamic and uncertain environments.
Table of Contents

List of Figures ix
List of Tables xii
Acknowledgments xiii

Chapter 1
Introduction 1
1.1 Motivation and Research Issues . . . . . . . . . . . . . . . . . . . 2
1.1.1 Situation Awareness: Understanding Sensor Data . . . . . . 2
1.1.2 Adaptive Decision-Making with Limited Information . . . . 6
1.2 Research Objectives and Contributions . . . . . . . . . . . . . . . 8
1.2.1 Supervised Learning from Time-Series Data . . . . . . . . . 8
1.2.2 Multi-Modal Fusion of Heterogeneous Sensors . . . . . . . . 9
1.2.3 Autonomous Navigation in Uncertain Environments . . . . . 11
1.3 Organization of the Dissertation . . . . . . . . . . . . . . . . . . . 12

Chapter 2
Supervised Learning from Time-Series Data:
A Symbolic Dynamic Approach 16
2.1 A Brief Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
2.1.1 Symbolic Dynamics . . . . . . . . . . . . . . . . . . . . . . . . 17
2.1.2 Symbolic Dynamic Filtering . . . . . . . . . . . . . . . . . . 18
2.2 Signal Pre-Processing . . . . . . . . . . . . . . . . . . . . . . . . . 19
2.2.1 Outlier Removal, Normalization and Missing Data . . . . . . 20
2.2.2 Data Transformation . . . . . . . . . . . . . . . . . . . . . . . 21
2.3 Symbolic Dynamics and Encoding . . . . . . . . . . . . . . . . . 23
2.4 Feature Extraction via Construction of Probabilistic Finite State Automata ........................................... 25
   2.4.1 Conversion from Symbol Image to State Image ................. 26
   2.4.2 State Compression .................................................... 26
   2.4.3 Construction of PFSA ................................................ 28
   2.4.4 Summary of SDF for feature extraction ............................ 29
2.5 Feature Selection and Pattern Classification .......................... 30
   2.5.1 Feature Selection ..................................................... 31
   2.5.2 Pattern Classification ................................................ 32

Chapter 3
Behavior Recognition in Mobile Robots 34
3.1 Introduction ................................................................. 34
3.2 Experimental Procedure ................................................ 36
3.3 Pattern Analysis for Behavior Recognition ............................ 39
3.4 Experimental Results and Discussion ................................. 41
3.5 Conclusions and Future Work .......................................... 45

Chapter 4
Anomaly Detection in Nuclear Power Plants 47
4.1 Motivation ........................................................................ 48
4.2 Data-Driven Anomaly Detection posed as a Multi-Class Pattern Classification Problem .............................................. 50
4.3 Optimization of Partitioning ............................................. 54
4.4 Anomaly Detection Algorithm ........................................... 58
   4.4.1 Symbolic Dynamic Filtering for Anomaly Detection ........... 59
   4.4.2 Principal Component Analysis for Anomaly Detection ...... 60
4.5 Results and Discussion ..................................................... 60
4.6 Conclusions and Future Work .......................................... 66

Chapter 5
Multi-Modal Sensor Fusion for Personnel Detection 69
5.1 Introduction ................................................................. 69
5.2 The Architecture of Multi-Level Fusion ............................... 71
5.3 Data Fusion - Cross Machine ........................................... 73
   5.3.1 Conversion of Symbol Images to Symbol Sequences ........ 73
   5.3.2 Construction of Cross Machine .................................... 75
5.4 Feature Fusion - Feature Selection .................................... 76
5.5 Validation on Border Security Data ................................... 78
   5.5.1 Data Collection and Preprocessing ................................. 78
## List of Figures

2.1 Symbol image generation via wavelet transform of the sensor time series data and partition of the wavelet surface in ordinate direction 24
2.2 Illustration of classical partition schemes (alphabet size $|\Sigma| = 3$) 25
2.3 Conversion of the symbol image to the state image 27
2.4 An example of a 4-state probabilistic finite state automaton (PFSA) 28
2.5 An example of feature extraction from the state image 29
2.6 Feature subset selection methods: wrapper and filter 31
2.7 Flow chart of the proposed methodology 33
3.1 Robot hardware: Pioneer 2AT and Segway RMP 36
3.2 Sensor layout in the laboratory environment 36
3.3 Example of sensor readings and plot of Haar wavelet 38
3.4 Ensemble mean of the state probability vectors (patterns) for pattern identification (a) Segway random; (b) Segway circle; (c) Segway square; (d) Pioneer random; (e) Pioneer circle; (f) Pioneer square 39
3.5 Tree structure for pattern identification 40
3.6 Classification error vs. neighbor size in $k$-NN classifiers 43
4.1 Class label assignment in the parameter space for data generation 52
4.2 Profile of turbine load and turbine output response 52
4.3 Representative time series data for RCP anomaly and $T_{HL}$ degradation conditions 53
4.4 General framework for optimization of feature extraction 57
4.5 Flow chart for data-driven anomaly detection 59
4.6 Two dimensional objective space for partitioning optimization 61
4.7 Feature space of the training set using optimal partitioning 62
4.8 Feature space of training set – uniform partitioning (UP) 62
4.9 Feature space of training set – maximum entropy partitioning (MEP) 62
4.10 Classification error vs. neighbor size in $k$-NN classifiers 65
5.1 Comparison of different sensor fusion methods: data fusion, feature fusion, decision fusion, and multi-level fusion ........................................ 72
5.2 An illustration of converting symbol images to symbol sequences ... 75
5.3 Cross machine constructed with symbol sequences in Fig. 5.2 .... 76
5.4 Feature selection using cascaded filter structure ...................... 78
5.5 Description of the experimental scenarios ............................ 79
5.6 Typical examples of sensor signals collected from the field test ... 81
5.7 Multi-level fusion of multi-modal sensors for personnel detection .... 82

6.1 Multi-resolution formulation of the search space ..................... 94
6.2 Illustration of the switch between local and global navigation..... 95
6.3 Mobile robots and its sensing and computation areas .............. 96
6.4 Flowchart of multi-resolution navigation ............................. 98
6.5 Illustration of the Bug2 algorithm .................................... 103
6.6 A screenshot of the Stage simulator .................................. 106

7.1 Typical examples of tasks that require complete coverage of unknown environments: (a) floor cleaning robot, (b) landmine removal using a mine flail in Sudan, and (c) naval mine countermeasure . . . 109
7.2 Example of complete coverage in a map with a maze and a large obstacle ................................................................. 115
7.3 Example of complete coverage in an apartment map ............... 116
7.4 Example of complete coverage in obstacle rich environment .... 117
7.5 Example of complete coverage in a map with large obstacles .... 118
7.6 The exogenous potential field $B_\xi^*$ designed for different searching patterns ................................................................. 119
7.7 Example of complete coverage in a spiral maze using zigzag pattern 120
7.8 Example of complete coverage in a spiral maze using spiral pattern 121
7.9 This history of percentage coverage in different experiments .... 122

8.1 The oil slick of Deepwater Horizon oil spill as seen from space by NASA's Terra satellite on May 24, 2010. The oil slick is brighter than the surrounding water in some places (image center) and darker than the surrounding water in others (image lower right) [168]. . . . . 125
8.2 Novel prototypes of autonomous systems for oil spill confrontation . 126
8.3 Spreading and drift of the oil spill due to wind and current .... 129
8.4 Oil spill cleaning in a harbor using the proposed multi-resolution algorithm ................................................................. 135
8.5 Oil spill cleaning in a harbor using the benchmark algorithm that has no adaptation to the oil spills ................................. 135
8.6 Oil spill cleaning profiles of the proposed and benchmark methods . 136

B.1 Layout of the primary side of the IRIS system [179] . . . . . . . . . . . . . . 147

C.1 Block diagram of the data fusion scheme . . . . . . . . . . . . . . . . . . . 150
C.2 Block diagram of the feature fusion scheme . . . . . . . . . . . . . . . . . . 150
C.3 Block diagram of the decision fusion scheme . . . . . . . . . . . . . . . . . 151
List of Tables

3.1 Parameters used for Various Types of Motion .......................... 37
3.2 Results of Robot and Motion Classification ............................. 42
3.3 Comparison of Computational Complexity of Feature Extraction Methods .................................................. 44

4.1 Anomaly Types in Reactor Coolant Pump (RCP) ....................... 51
4.2 Noise Level and SNR in the $T_{HL}$ Sensor .............................. 51
4.3 Comparison of Classification Performances of Different Methods on Testing Data Set (50 × 9 Samples) .............................................. 66

5.1 Number of Feature Vectors for Each Target Class in the Data Sets . 80
5.2 Comparison of the Classification Accuracy of using Different Methods 84

7.1 Quantitative Evaluation of the Proposed Algorithm in Different Experiments ......................................................... 122
8.1 Comparative Evaluation of the Proposed and Benchmark Methods in Oil Spill Cleaning in a Harbor Environment ................................. 136
Acknowledgments

This thesis would not have been possible without the help of many wonderful people with whom I have had the pleasure to work throughout these years. First and foremost, I would like to thank my advisor Dr. Asok Ray. He has been an invaluable source of inspiration and encouragement since the day I arrived in State College. Without his moral and academic support, I would not have been able to overcome the challenges I faced during this journey. The other members of my thesis committee were also instrumental in allowing me to successfully achieve this goal. I would like to thank Dr. Alok Sinha, Dr. Shashi Phoha, Dr. Jeffrey Mayer, and Dr. Thomas Wettergren for serving on my committee and providing valuable suggestions to improve my dissertation.

I would also like to thank Dr. Thyagaraju Damarla for sharing his expert knowledge in personnel detection, and Dr. Robert Edwards for his technical support in my research work in nuclear engineering. I would like to extend my thanks to my colleagues at Penn State with whom I had a wonderful experience as a graduate student. I would especially like to thank Dr. Soumik Sarkar and Dr. Kushal Mukherjee for their support in my research work. I would like to thank Dr. Shalabh Gupta, Dr. Abhishek Srivatsav, Dr. Eric Keller, Dr. Goutham Mallapragada, Dr. Subhadeep Chakraborty, Dr. Chinmay Rao, Dr. Yicheng Wen, Mr. Anthony Cascone, Ms. Jacqueline Luff, Mr. Soumalya Sarkar, Mr. Nurali Virani, and all other members of Dr. Ray’s group for their help on various occasions.

I would like to thank my parents and Ms. Ying Shi for their love and support along the way. I am deeply indebted to them for their dedication, encouragement and inspiration without expecting anything in return.

Lastly, I would like to gratefully acknowledge the financial support I received throughout these years: the U.S. Army Research Laboratory and the U.S. Army Research Office Grant No. W911NF-07-1-0376, the U.S. Office of Naval Research Grant No. N00014-09-1-0688, NASA Cooperative Agreement No. NNX07AK49A, and the U.S. Department of Energy NERI-C Grant No. DE-FG07-07ID14895.
Dedication

Dedicated to Ying and my parents
Chapter 1

Introduction

Advances in technology especially the advent of computer technology in the late 20th century have redefined the lifestyle of human society. With artificial intelligence, machines exhibit certain levels of autonomy in perception, learning, and decision-making. These machines are referred to as autonomous systems in the sequel. Autonomous systems are becoming prevalent in many areas such as planetary exploration, military operations, industrial manufacturing, and medical systems. Autonomous systems play a very important role in the applications where the systems need to function in harsh or hermetic environments, such as the implantable devices inside the human body where it is impossible to use wires. The Curiosity Rover that recently landed on Mars is another typical example of autonomous systems. Due to the bandwidth and delay in communication between the rover and mission control on Earth, the rover has to perform many tasks such as path planning and sample analysis on its own without human intervention.

There are three commonly accepted primitives of autonomous systems: sensing, planning and acting [1]. Control of autonomous systems can be categorized based on the relationship between these three primitives and in terms of how sensory data are processed and propagated. The autonomous systems use sensors of various modalities to percept the environment and generate relevant information. In this process, the autonomous system extracts knowledge from the generated information and assesses the situation in the environment. With the extracted knowledge, the autonomous system performs certain kind of reasoning, compares the outputs with its predefined goals, and makes plans about how to achieve these
goals. Finally, the autonomous system executes its plan by using its onboard actuators, whose type depends on the nature of the task and the operating environment. After achieving a goal, the autonomous system makes new plans to achieve the remaining goals. It is also possible that the autonomous system aborts the current mission and switches to another mission that has higher priority, upon discovery of new knowledge from sensor data. Therefore, perception of the environment and decision-making based on the extracted knowledge are crucial for successful operation of autonomous systems.

1.1 Motivation and Research Issues

Common applications of autonomous systems typically fall under sensor-based systems and decision-making systems. The major difference between these two systems is that the former extracts the knowledge from sensory data and relies on human operator for decision-making, whereas the latter implements the decision-making process based on the generated information. The advent of modern sensors and sensing applications has led to the capability of generating large volumes of data. Extraction of useful information from these data sets, in real time, has enhanced the ability to better comprehend and analyze the environment. Decision-making systems rely on the generated information from the sensor-based system for planning the actions. Due to limitation of the onboard sensors as well as complexity and uncertainties of the environment, the decision-making systems may not have full knowledge of the environment and thus need to make decisions based on the available information.

1.1.1 Situation Awareness: Understanding Sensor Data

Situation awareness is the perception of environmental elements with respect to time and/or space, the comprehension of their meaning, and the projection of their status after some variable has changed, such as time. It is also a field of study concerned with perception of the environment critical to decision-makers in complex, dynamic areas from aviation, air traffic control, power plant operations [2], military command and control [3], and emergency services; to more ordinary but
nevertheless complex tasks such as driving an automobile or bicycle [4].

Autonomous systems are required to perceive the environments by sensors, extract useful information from the sensor data in real time, and send the abstract feedback information to the end-users. For example, an autonomous system that contains a network of surveillance cameras is often present in cities, aimed at maintaining a comprehensive coverage of specific locations such as a building or other sensitive areas. The main goal of this system is to assist the people to detect and identify potential threats or suspicious events during the recorded time frame of the video. Since it is inefficient for human operators to watch the monitors for anomalous situations on a 24/7 basis, the autonomous system that analyzes the video data by using an anomaly detection algorithm comes into play. By analyzing the video data continuously and sending alerts to human operators only when suspicious events are detected, the autonomous systems significantly increase the efficiency of information processing and thus reduces the load of the human operator. Therefore, it is crucial to implement machine learning algorithms for situation awareness in sensor-based systems.

Tools of data-driven feature extraction and pattern classification facilitate performance monitoring of distributed dynamical systems over a sensor network, especially if the physics-based models are either inadequate or unavailable. In this regard, a critical issue is real-time analysis of sensor time series for information compression into low-dimensional feature vectors that capture the relevant information of the underlying dynamics [5, 6, 7, 8].

The sensors that are used to perceive the environments can be broadly categorized into imaging sensors and non-imaging sensors. Both types of sensors are widely used in different applications. The imaging sensors provide intuitive understanding of the environments at the expense of storage, power consumption and processing speed. Non-imaging sensors, which provide time-series data as the observations of the environments, are more widely used in the occasions where on-board computation capability and power source are constrained. The latter case is much more common in the real-world environments, and thus this dissertation focuses on analysis of time-series data from non-imaging sensors.

**Time Series Analysis**
In general, there are two different data models of time series: numeric and symbolic times series [9]; a numeric time series can be converted into a symbolic time series by partitioning the phase space [10].

Time series analysis is a challenging task if the data set is voluminous (e.g., collected at a fast sampling rate), high dimensional, and noise contaminated. Moreover, in a distributed sensor network, data collection occurs simultaneously at multiple nodes. Consequently, there is a need for low-complexity algorithms that could be executed locally at the nodes to generate compressed features and therefore reduce the communication overhead. In general, the success of data-driven pattern classification tools depends on the quality of feature extraction from the observed time series. To this end, several feature extraction tools, such as principal component analysis (PCA) [7], independent component analysis (ICA) [11], kernel PCA [12], dynamic time warping [13], derivative time-series segment approximation [14], artificial neural networks (ANN) [15], hidden Markov models (HMM) [16], and wavelet transforms [17, 18, 19] have been reported in technical literature. Wavelet packet decomposition (WPD) [17] and fast wavelet transform (FWT) [18] have been used for extracting rich problem-specific information from sensor signals. Feature extraction is followed by pattern classification (e.g., using support vector machines [SVM]) [5, 6].

The concepts of Symbolic Dynamics [10, 20] have been used for information extraction from time series in the form of symbol sequences [21][22]. Keller and Lauffer [23] used tools of symbolic dynamics for analysis of time series data to visualize qualitative changes of electroencephalography (EEG) signals related to epileptic activity. Along this line, a real-time data-driven machine learning tool, called symbolic dynamic filtering (SDF) [10][24], has been built upon the concepts of symbolic dynamics and information theory. In the SDF method, time series data are converted to symbol sequences by appropriate partitioning [24]. Subsequently, probabilistic finite-state automata (PFSA) [10] are constructed from these symbol sequences that capture the underlying system’s behavior by means of information compression into the corresponding state-transition probability matrices. SDF-based pattern identification algorithms have been shown by experimental validation in the laboratory environment to yield superior performance over several existing pattern recognition tools (e.g., PCA, ANN, particle filtering, unscented
Kalman filtering, and kernel regression analysis \([6][5]\)) in terms of early detection of small changes in the statistical characteristics of the observed time series \([25]\).

Partitioning of time series is a crucial step for symbolic representation of sensor signals. To this end, several partitioning techniques have been reported in literature, such as *symbolic false nearest neighbor partitioning* (SFNNP) \([26]\), *wavelet-transformed space partitioning* (WTSP) \([27]\), and *analytic signal space partitioning* (ASSP) \([28]\). In particular, the wavelet transform-based method is well-suited for time-frequency analysis of non-stationary signals, noise attenuation, and reduction of spurious disturbances from the raw time series data without any significant loss of pertinent information \([29]\). In essence, WTSP is suitable for analyzing the noisy signals, while SFNNP and ASSP may require additional preprocessing of the time series for denoising. However, the wavelet transform of time series introduces two new domain parameters (i.e., scale and shift), thereby generating an image of wavelet coefficients. Thus, the (one-dimensional) time series data is transformed into a (two-dimensional) image of wavelet coefficients. In WTSP, the conversion of the image of the wavelet coefficients to one-dimensional domain of scale-series may be non-unique and lossy \([28]\).

**Multi-Modal Sensor Fusion**

Another important issue in situation awareness is the fusion of sensor data. Autonomous systems rely on various sensors to acquire information from the environment. A single sensor has its limitations in many aspects \([30]\), such as limited sensing coverage, vulnerable to noise, lack of robustness. There are many instances where it is not possible for a single sensor or information source to derive inferences adequately. For this reason, multi-modal sensor fusion comes into picture. The sensors used in fusion can be either complementary, competitive or cooperative \([31]\). The complementary sensors do not depend on each other directly but they are capable of providing a complete picture of the entire region. The competitive sensors deliver independent measurement of the same attribute or feature, and provide robustness and fault-tolerance. The cooperative sensors provide data that are used to derive information that would not be available from a single sensor, such as in the stereoscopic vision system. More details about the type of sensor configuration are given in Appendix C.2.
Over the decades, tools of multi-modal sensor fusion have been developed to solve a diverse set of problems with common characteristics. It seeks to combine the results of multiple sensors so that more accurate and robust information can be obtained and more reliable conclusions can be drawn. Many existing approaches exist in the literature of speaker detection and tracking where the correlation between a microphone signal and a camera image is studied using neural network [32], Bayesian network [33], particle filter [34], and information theoretic [35] approaches.

A parametric framework for fusing information from heterogeneous sources was introduced by Iyengar et al. [36], where a copula-based method was used to incorporate statistical dependence between disparate sources of information. The problem of identifying the best copula for binary classification problems was also addressed in [36]. However, the copula-based method assumes knowledge of the probability density function (pdf) of random vectors, which may not be available in many applications. Although it is possible to estimate the pdf’s by statistically fitting the observed data [37], the accuracy of the estimation depends on the selection of statistical distribution and the goodness of fitting to the data.

Information fusion in a sensor network is deemed critical for both military and civilian applications and has found relevance in various applications such as tactical plan recognition [38], battlefield situation awareness problem [39], threat evaluation in air defense scenarios [40], and disaster response [41].

1.1.2 Adaptive Decision-Making with Limited Information

Autonomous systems usually operate in partially or completely unknown environments, and need to replan its actions online based on information extracted from the environments. Therefore, decision-making in an autonomous system should be adaptive based on the available information about the environment, and thus needs to be updated when new information becomes available.

A large number of applications in decision-making in autonomous systems fall under the category of path planning and navigation of autonomous systems. In this dissertation, the adaptive decision-making problem is addressed in the context of path planning and navigation. Path planning and navigation for autonomous
systems, in the particular case where the environment is known, is a well studied problem (see, for example, the book by Latombe [42] and the references therein). In practice, however, complete knowledge about the environment is often unavailable; For example, having a detailed map with all the obstacles marked seems to be unrealistic for most situations. In many outdoor applications the robots can determine their coordinates by using GPS. However, the knowledge about the environment may often be very limited; under such conditions there is ample uncertainty for a very detailed plan to make sense. It is important for an autonomous system to be able to replan the path online based on the new information generated by sensors while navigating [43].

A natural way of updating plans is to first select a path based on the present knowledge, then move along that path for a short time while collecting new information. Based on the new findings the path is then replanned. This methodology is often used in the literature for path planning in unknown areas [43]. One of the original motivations for studying this problem was the terrain acquisition problem, where an autonomous system is required to produce a complete map of an unknown terrain. In many publications graph methods are used for solving the task [43, 44, 45]. The most well-known path planning algorithms are A* and its dynamic variant, i.e., D* [46, 47], and ν* [48].

The path planning problem becomes more complex when the autonomous system is required to scan all the points in the specified search area, and this problem is known as complete coverage problem. In many applications, the autonomous systems are not provided with a map, and they have to construct a map for path planning based on the available sensor information. Typical applications of complete coverage include humanitarian demining [49], floor cleaning [50] and terrain map generation [51]. Arkin et al. [52] have shown that finding the optimal coverage path in a known map is an NP-hard problem. They proved this condition by formulating the coverage as a traveling salesman problem, with each location in the reachable area as a city. It is important to use an efficient algorithm for coverage planning with the available information and update the navigation plan as new information becomes available. Recent coverage algorithms for operating in unknown environments are based on Morse decomposition (i.e., cellular decompositions of the search space using critical points of the Morse functions) [53][54][55]
and slice decomposition [56].

1.2 Research Objectives and Contributions

Based on the motivation and research issues discussed in the previous section, the objectives and contributions of this dissertation are broadly divided into two parts that are delineated in the sequel. The first part discusses the research in situation awareness. The second part discusses adaptive decision-making with applications in autonomous navigation in uncertain environments.

Part I: Situation Awareness

Part I presents two topics of situation awareness: supervised learning from time-series data (Sec. 1.2.1) and multi-modal fusion of heterogeneous sensors (Sec. 1.2.2).

1.2.1 Supervised Learning from Time-Series Data

The sensor information obtained from the environment needs to be analyzed for the purpose of identifying small changes for local adaptation or large changes for replanning. From the consideration of autonomy, it is important that the low power sensing platforms are able to perform in-situ signal processing and pattern recognition analysis. Data transformation is usually performed to denoise or improve the interpretability of the embedded information in the transformed space. However, the small sensing platforms may not have the capability to perform data transformation due to the limitation of computation power. In this circumstance, algorithms that can effectively extract the pertinent information from noise-contaminated sensor measurement is crucial for the purpose of situation awareness. In this dissertation, learning and analysis of time-series data is formulated in a supervised manner. The class label information is provided when signal processing and pattern recognition algorithms are used to process the training data. Supervised learning is one of the most common approaches, especially in the applications of behavior recognition, fault diagnosis, and target identification.

The data models of time series can be broadly divided into two categories: numeric and symbolic [9]. A numeric time series can be converted into a symbolic time series by partitioning the phase space [10]. The symbolic time series
have several advantages over the numeric time series such as low computational requirement, robustness to noise and ease of storage and distribution. Recent literature shows a promising way of extracting knowledge from symbolic time series via formal linguistic representations, known as probabilistic finite state automata (PFSA) [57]. As discussed in Section 2, a probabilistic finite state automaton (PFSA) is essentially a finite-state machine with defined event generation probabilities. The processes of transforming time series data to symbolic sequences over abstract alphabets and constructing PFSA from symbolic sequences are formulated in the framework of symbolic dynamic filtering (SDF) [10, 25]. As one may infer from the description in this section, data transformation, symbolization of the time series data, and extraction of pertinent information from the symbol sequences are the three crucial steps in SDF. Many papers that address these research topics can be found in literature [28, 27, 58].

This dissertation focuses on these crucial steps in SDF and extending the previous work to extract information from noisy time-series data in a more effective manner. To improve the quality of feature extraction, various signal pre-processing techniques are investigated, such as Hilbert transform [28, 58] and wavelet transform [27]. However, Hilbert transform is not particularly effective for noise-contaminated time-series data, and the feature extraction from the two-dimensional scale and shift domain of the wavelet-transformed data may be lossy and non-unique. This dissertation extends the concept of SDF for feature extraction in the two-dimensional scale-shift domain of wavelet transform without any need for non-unique conversion to one-dimensional sequence. Although modeling of state machines from symbol sequences is widely reported, similar efforts have not been expended to investigate data partitioning to optimally generate the symbol sequences. This dissertation addresses this issue and proposes a data partitioning procedure to extract low-dimensional features from time series while minimizing the loss of class separability information.

1.2.2 Multi-Modal Fusion of Heterogeneous Sensors

Due to possible multi-modality of sensors, the relevant information need to be fused effectively in-situ to capture the emerging patterns of interest. Depending
on levels where the fusion is performed, sensor fusion techniques can be categorized into three types: data fusion, feature fusion, and decision fusion. Data fusion has the least amount of information loss, but may be vulnerable to measurement noise and loss of synchronization. Fusion of heterogeneous sensors at data level is more difficult than homogeneous sensors because the characteristics of the sensor signals may be significantly different. Decision fusion is less sensitive to measurement fluctuations and sensor types, but may suffer from information loss because the fusion is performed at the highest level.

Although feature extraction from symbol sequences from a single sensor has been widely studied, fusion of symbolic time series from heterogeneous sensor received much less attention. One way to fuse the symbolic time series from multiple sensors is to map the sensor readings to a multi-dimensional phase space where each dimension is partitioned into coarse grains. However, partitioning a multi-dimensional phase space is a non-trivial task, especially when non-uniform partitioning methods are used. The same symbol may appear in different locations in the phase space if the partitioning of the multi-dimensional phase space is generated in different orders. To circumvent this problem, Srivastav [59] introduced the concept of cross machine for object and situation assessment in sensor networks, and Sarkar et al. applied this concept to fault diagnosis in aircraft gas turbine engine [60]. However, the dimension of feature vectors generated by cross machine is the product of the alphabet size of one symbol sequence and the number of state of the other symbol sequence. The feature vectors may be sparse since many states may not be visited, and thus many features become redundant and do not provide any class separability information.

This dissertation introduces the framework of multi-level fusion for extraction of the relational information between heterogenous sensors in an efficient way. As the first step in the proposed framework, cross machine is used to generate the relational patterns from each sensor pairs. Feature selection is then performed to select a subset of the representative features, which have much smaller size than the original feature set and yet carry most of the information.

Therefore, the main contributions of Part I of this dissertation are:

• Improved wavelet-based feature extraction from time-series data
Part I (Chapters 2, 3, 4 and 5) presents these contributions in detail along with validation examples and they are summarized as well in Chapter 9.

Part II: Adaptive Decision-Making

1.2.3 Autonomous Navigation in Uncertain Environments

This dissertation addresses the issue of adaptive decision-making in autonomous systems in the context of autonomous navigation in uncertain environments. The autonomous system operates in a priori unknown or partially known environment and carries out its task. Due to the limited sensing capability, the autonomous system only has partial information about the environment. The autonomous system needs to make navigation decisions in real time based on the available information and updates its navigation plan as new information about the environment is obtained by its sensors.

This dissertation presents a symbolic approach to solve the problem of adaptive decision-making. First of all, the search space is formulated into a symbolic grid map. The symbolic grid map is generated by partitioning the two-dimensional search space into a fine grid and then assigning an alphabet of symbols to each grid cell to represent the obtained information about the environment. Formulation of the search space is then extended in the multi-resolution sense to enable adaptive decision-making based on the available spatio-temporal information. The symbolic grid map is updated when the previously unexplored part of the search space is revealed by the onboard sensors of the autonomous system. The proposed algorithm has low computational complexity and does not suffer from the local minima problem. The proposed algorithm is validated in various scenarios where the autonomous systems carry out different tasks in complex and dynamic environments.

Therefore, the main contributions of Part II of this dissertation are:

• Multi-resolution formulation of the search space that enables adaptive decision-
making

- Development of algorithms for navigation of autonomous systems in complex and dynamic environments
- Experimental validation of the multi-resolution algorithm in various scenarios

Part II (Chapters 6, 7 and 8) presents these contributions in detail along with validation examples and they are summarized as well in Chapter 9.

1.3 Organization of the Dissertation

The dissertation is divided into two parts based on the research objectives described in the previous section. Part I encompasses the contribution made in the area of situation awareness from sensor time-series data using a symbolic dynamic approach. Part II presents the contribution made in the area of autonomous navigation in uncertain environments by adaptively learning the environments in a symbolic way. These two parts are self-contained and are also correlated. Part I presents the ways that an autonomous system uses to percepts the environment and assesses the situation, while Part II describes how the autonomous system reacts based on the information learned from the environment. Brief descriptions of the chapters of Part I are provided below, followed by those of Part II.

Part I: Situation Awareness: Understanding Sensor Data

- Chapter 2 presents a framework of supervised learning from sensor time-series data using a symbolic dynamic approach. This chapter first reviews the concepts of symbolic dynamic filtering (SDF), and then extends the concept of SDF for feature extraction in the two-dimensional scale-shift domain of wavelet transform without the need for non-unique conversion to one-dimensional sequence. Feature selection and pattern classification algorithms are also presented as part of the framework of supervised learning.

- Chapter 3 presents the experimental validation of the wavelet-based feature extraction method proposed in Chapter 3. The objective of this chapter is to identify the robot type and the behavior (i.e., the motion type) based on the time series data obtained from the piezoelectric pressure sensitive floor.
The performance of the proposed method is compared with that of principal component analysis (PCA) by using standard pattern classification tools. The complexity of the proposed method is also evaluated in this chapter.

• Chapter 4 proposes an anomaly detection algorithm for condition monitoring of nuclear power plants, where symbolic feature extraction and the associated pattern classification are optimized by appropriate partitioning of (possibly noise-contaminated) sensor time series. In this process, the system anomaly signatures are identified by masking the sensor degradation signatures. The task of anomaly detection is viewed as a multi-class pattern classification problem, where feature extraction is optimized to enhance the classification rate. The proposed anomaly detection methodology is validated on the International Reactor Innovative & Secure (IRIS) simulator of nuclear power plants, and its performance is evaluated by comparison with that of PCA.

• Chapter 5 focuses on fusion of heterogeneous sensors on unattended ground sensor (UGS) systems that are used to monitor human activities. Existing algorithms have encountered difficulties due to intolerably high rates of data transfer and misclassification. The objective of this chapter is to develop an efficient algorithm that is able to distinguish a human from a large animal with high classification accuracy. Traditional fusion techniques are reviewed in this chapter, and then a multi-level fusion algorithm is introduced to combine the advantages of data fusion and feature fusion. The proposed multi-level fusion algorithm is validated on a data set collected from the test fields near the U.S. southern border, and its performance is compared with several benchmark techniques in personnel detection.

Part II primarily deals with the problem of autonomous navigation in uncertain environments. The environment is represented by a symbolic grid map that is partitioned into a mutually exclusive and exhaustive set of cells, where each cell stores the information about the search space in a symbolic way. A multi-resolution algorithm is then introduced to mimic the adaptive decision-making process in biological systems on the symbolic grid map. Various scenarios in both static and dynamic environments are studied in this part to demonstrate the capability of
the proposed multi-resolution algorithm for autonomous navigation in uncertain environments.

Part II: Adaptive Decision-Making:
Autonomous Navigation in Uncertain Environments

- Chapter 6 presents the problem statement of autonomous navigation in uncertain environments. The environment is formulated into a symbolic grid map where each grid represents the autonomous system’s knowledge about the environment using an alphabet of symbols. These symbols are updated by the sensory information about the environment. A multi-resolution algorithm is then proposed for autonomous navigation using the symbolic grid map. The proposed algorithm seamlessly integrates the concepts of local navigation and global navigation, and switches between these two modes depending on the amount of spatio-temporal information needed for making the navigation decision. These notions are inspired from biological systems that make instantaneous decisions based on the local sensory information and the long-distance decisions based on (partially) available global information.

- Chapter 7 presents an application of the multi-resolution algorithm proposed in Chapter 6 for complete coverage of static and complex environments. Additional constraints are applied to the multi-resolution algorithm to improve the efficiency of the algorithm in exploration and reduces the numbers of unnecessary turns and overlapping trajectory. The algorithm is tested in various complex scenarios to show the robustness of the algorithm using a high-fidelity Player/Stage simulator. Performance metrics are also introduced to quantitatively evaluate the effectiveness and efficiency of the algorithm in implementing complete coverage of the uncertain environment.

- Chapter 8 presents the problem of oil spill cleaning in uncertain environments by an autonomous system. This topic is inspired by the recent Deepwater Horizon oil spill incident in the Gulf of Mexico. The application of the multi-resolution algorithm proposed in Chapter 6 is extended to dynamic and uncertain environments due to the spreading and drift of the oil spills. The 2D random-walk particle-tracking technique is used to follow the motion
of individual particles (oil droplets), the total amount of which constitutes the oil spill. The algorithm is validated on the Player/Stage simulator with a scenario of oil spill cleaning in a harbor. Several performance metrics are proposed to compare the performance of the multi-resolution algorithm with that of the benchmark method, which uses a zigzag motion pattern to cover the environment.

The dissertation is summarized and concluded in Chapter 9 along with a few recommendations of future research directions that can possibly emerge from this research. Finally, the dissertation is supported by four appendices.

- Appendix A describes the principal component analysis (PCA) algorithm that is used as the benchmark algorithm of feature extraction for comparison with SDF in both Chapter 3 and Chapter 4.

- Appendix B describes the IRIS simulator of nuclear power plants, which is used for validation in Chapter 4.

- Appendix C provides an overview of some of the existing techniques of information fusion. Three levels of information fusion, namely, data fusion, feature fusion, and decision fusion are described in this appendix.

- Appendix D presents a brief review of the oil spill modeling and cleaning methods that serves as the background for part of the research presented in Chapter 8.
Chapter 2

Supervised Learning from Time-Series Data: A Symbolic Dynamic Approach

Machine learning is the science concerned with the design and development of algorithms, which allows computers to identify evolving behaviors based on sensor data. Machine learning algorithms are organized into a taxonomy based on the outcome of the algorithm. The major categories include supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Supervised learning generates a function that maps inputs to desired outputs (i.e., class labels). Unsupervised learning models a set of inputs, like clustering. Semi-supervised learning combines both labeled and unlabeled examples to generate an appropriate classifier. Reinforcement learning is concerned with how an agent ought to take actions in an environment so as to maximize a notion of cumulative reward [61].

A recently developed statistical pattern recognition tool, called Symbolic Dynamic Filtering (SDF), is built upon the concepts from multiple disciplines including Symbolic Dynamics [10] and Information Theory [62]. It is shown to be a fast, efficient and computationally inexpensive pattern recognition tool [25, 63]. These qualities are particularly important for situation awareness applications as discussed earlier in the introduction chapter. This chapter proposes a framework for supervised learning from time-series data, where SDF serves as a feature extraction
tool that generates feature vectors for pattern classification.

2.1 A Brief Overview

Real-time data-driven pattern classification requires extraction of relevant features from the observed time series as low-dimensional and yet information-rich representations of the underlying dynamics. These low-dimensional features facilitate in-situ decision-making in diverse applications, such as computer vision, structural health monitoring, and robotics. Symbolic dynamics-based methods, have been developed to capture the relevant information that is embedded in the sensor data. This section first briefly describes the concept of symbolic dynamics, and then presents an overview of the SDF framework.

2.1.1 Symbolic Dynamics

Symbolic Dynamics [20] is the practice of modeling a dynamical system by a discrete space consisting of sequences of abstract symbols, each of which corresponds to a state of the system, with the dynamics (evolution) given by the shift operator. The concepts of Symbolic Dynamics have been used for information extraction from time series in the form of symbol sequences [21][22]. Keller and Lauffer [23] used tools of symbolic dynamics for analysis of time series data to visualize qualitative changes of electroencephalography (EEG) signals related to epileptic activity.

In the symbolic dynamics literature [20], it is assumed that the observed sensor time series from a dynamical system are represented as a symbol sequence. Let \( \Omega \) be a compact (i.e., closed and totally bounded) region in the phase space of the continuously-varying dynamical system, within which the observed time series is confined [10][24]. The region \( \Omega \) is partitioned into \(|\Sigma|\) cells \( \{\Phi_0, \cdots, \Phi_{|\Sigma|-1}\} \) that are mutually exclusive (i.e., \( \Phi_j \cap \Phi_k = \emptyset \quad \forall j \neq k \)) and exhaustive (i.e., \( \bigcup_{j=0}^{|\Sigma|-1} \Phi_j = \Omega \)), where \( \Sigma \) is the symbol alphabet that labels the partition cells. A trajectory of the dynamical system is described by the discrete time series data as: \( \{x_0, x_1, x_2, \cdots\} \), where each \( x_i \in \Omega \). The trajectory passes through or touches one of the cells of the partition; accordingly the corresponding symbol is assigned to each point \( x_i \) of the trajectory as defined by the mapping \( \mathcal{F} : \Omega \to \Sigma \). Therefore, a
sequence of symbols is generated from the trajectory starting from an initial state $x_0 \in \Omega$, such that

$$x_0 \rightarrow \sigma_0 \sigma_1 \sigma_2 \ldots \sigma_k \ldots$$

where $\sigma_k \triangleq F(x_k)$ is the symbol at instant $k$. (Note: The mapping in Eq. (2.1) is called Symbolic Dynamics if it attributes a legal (i.e., physically admissible) symbol sequence to the system dynamics starting from an initial state).

### 2.1.2 Symbolic Dynamic Filtering

In the SDF method, time series data are converted to symbol sequences by appropriate partitioning [24]. Subsequently, a probabilistic finite-state automata (PFSA) [10] is constructed from these symbol sequences that capture the underlying system’s behavior by means of information compression into the corresponding state-transition probability matrices. SDF-based pattern identification algorithms have been shown by experimental validation in the laboratory environment to yield superior performance over several existing pattern recognition tools (e.g., PCA, ANN, Bayesian Filtering, Particle Filter, Unscented Kalman Filtering, and Kernel Regression Analysis) in terms of early detection of small changes in the statistical characteristics of the observed time series [25].

Partitioning of time series is a crucial step for symbolic representation of sensor signals. To this end, several partitioning techniques have been reported in literature, such as symbolic false nearest neighbor partitioning (SFNNP) [26], wavelet-transformed space partitioning (WTSP) [27], and analytic signal space partitioning (ASSP) [28]. In particular, the wavelet transform-based method is well-suited for time-frequency analysis of non-stationary signals, noise attenuation, and reduction of spurious disturbances from the raw time series data without any significant loss of pertinent information [29]. In essence, WTSP is suitable for analyzing the noisy signals, while SFNNP and ASSP may require additional preprocessing of the time series for denoising. However, the wavelet transform of time series introduces two new domain parameters (i.e., scale and shift), thereby generating an image of wavelet coefficients. Thus, the (one-dimensional) time series data is transformed into a (two-dimensional) image of wavelet coefficients. In this context, for SDF-based analysis of wavelet-transformed data, the prior work [10][27] suggested stack-
encoding of the wavelet coefficients from multiple scales to represent the two-dimensional scale-shift wavelet domain by a one-dimensional data sequence. However, this procedure of conversion of the two-dimensional domain to one-dimensional is non-unique and potentially lossy depending on the choice of the stacking procedure.

This chapter extends the concept of SDF for feature extraction in the (two-dimensional) scale-shift domain of wavelet transform without any need for non-unique conversion to one-dimensional sequence. In addition, the proposed method is potentially applicable for analysis of regular images for feature extraction and pattern classification. The major steps of the SDF method for feature extraction are delineated as follows [64]:

1. Encoding (possibly nonlinear) system dynamics from observed sensor data (e.g., time series and images) for generation of symbol sequences;

2. Information compression via construction of probabilistic finite state automata (PFSA) from the symbol sequences to generate feature vectors that are representatives of the underlying dynamical system’s behavior.

The remaining of this chapter is organized into three sections. Section 2.2 introduces several signal pre-processing methods as the first step of SDF. Section 2.3 describes the conversion of time-series data to symbol sequences. Section 2.4 presents feature extraction via construction of probabilistic finite-state automata (PFSA) from the symbol sequences. Section 2.5 presents the selection and classification procedures of the extracted features.

### 2.2 Signal Pre-Processing

Signal pre-processing is the first step of most signal processing techniques. The goal of signal pre-processing is to extract relevant information from the sensor data and prepare the data for pattern analysis. This section briefly reviews the commonly used signal pre-processing methods. Pre-processing methods that deal with the issues of outlier removal, data normalization, and missing data are reviewed in Sec. 2.2.1. Transformation of data into another domain may also be considered to be a pre-processing method, and it is discussed in Sec. 2.2.2.
2.2.1 Outlier Removal, Normalization and Missing Data

An outlier is defined as a point that lies very far from the mean of the corresponding random variable. The distance is measured with respect to a given threshold, usually a number of times the standard deviation. Points with values very different from the mean value produce large errors during the training stage of a machine learning technique and may have disastrous effects. For example, the least squares criterion is very sensitive to outliers, because large errors dominate the cost function due to the squaring of the terms [65]. A survey of outlier detection techniques is given in [66], and a review of related techniques that attempt to address such problems is given in [67].

In many practical situations, the data recorded by sensors may lie within different dynamic ranges. On one hand, detection of changes in the sensor data is very important in many applications, such as fault detection and isolation; on the other hand, data sets with different ranges may adversely affect the extraction of pertinent information. For example, in the target classification problem, the recorded acoustic signals by microphones depends on the strength of the signal source and the distance between the sensor and the target. Without normalization, the same target passing by the sensor site at different distances may appear to be different types of target in the feature space. The problem is overcome by normalizing the raw data such that their values lie within similar ranges. A straightforward technique is normalization via the respective estimates of the mean and variance.

In practice, certain data points may be missing due to the malfunction of data acquisition tools. The most traditional techniques in dealing with missing data include schemes that “complete” the missing values by (a) zeros or (b) the unconditional mean, computed from the available values of the respective data or (c) the conditional mean, if one has an estimate of the probability density function of the missing values given the observed data. Another approach is to simply discard data sample with missing values. Although such an approach can be useful in cases of large data sets, in most cases it is considered a “luxury” to afford to drop available information. Advanced techniques to handle missing data include Bayesian inference and logistic regression [68].
2.2.2 Data Transformation

Data transformation refers to the application of a deterministic mathematical function to each point in a data set. Transforms are usually applied so that the data appear to more closely meet the assumptions of a statistical inference procedure that is to be applied or to improve the interpretability of the data. In signal processing and machine learning problems, data transforms are performed to enhance the signature of the relevant information embedded in the signal or to improve the separability of the features among different classes in the transformed domain.

Hilbert transform and associated concept of analytic signals, introduced by Gabor [69], have been widely adopted for time-frequency analysis in diverse applications of signal processing. In Hilbert transform, the observed real-valued data sequence into the corresponding complex-valued analytical signal. A typical application of Hilbert transform is ECG signal analysis [70]. Generalization of Hilbert transform is explored by Sarkar et al. [58] to address symbolic analysis of noise-corrupted dynamical systems.

Fourier transform, perhaps the most well-known signal analysis tool, breaks down a signal into constituent sinusoids of different frequencies. In Fourier transform, time information is lost in transforming to the frequency domain. However, most interesting signals contain numerous non-stationary or transitory characteristics that are often the most important part of the signal. Short-Time Fourier Transform (STFT) uses a windowing technique to map a signal into a two-dimensional function of time and frequency. However, the time and frequency information provided by STFT has limited precision that is determined by the size of the window. Wavelet transform represents the next logical step: a windowing technique with variable-size regions. Wavelet analysis allows the use of long time intervals where precise low frequency information is required, and shorter regions where high frequency information is needed [71].

Wavelet transform can be performed for every scale and translation, resulting in continuous wavelet transform (CWT), or only at multiple scales and translation intervals, resulting in discrete wavelet transform (DWT) [19]. CWT is used in this dissertation is used as the data transformation tool owing to its time-frequency localization property.

In CWT, time series are first transformed into the wavelet domain, where
wavelet coefficients are generated at different shifts and scales. The choice of the wavelet basis function and wavelet scales depends on the time-frequency characteristics of individual signals [24]. The wavelet transform of a function \( f(t) \in \mathbb{H} \) is given by

\[
F_{s,\tau} = \frac{1}{\sqrt{\alpha}} \int_{-\infty}^{\infty} f(t) \psi_{s,\tau}^{*}(t) dt,
\]

where \( s > 0 \) is the scale, \( \tau \) is the time shift, \( \mathbb{H} \) is a Hilbert space, \( \psi_{s,\tau}(t) = \psi\left(\frac{t-\tau}{s}\right) \) and \( \psi \in L_{2}(\mathbb{R}) \) is such that \( \int_{-\infty}^{\infty} \psi(t) dt = 0 \) and \( ||\psi||_{2} = 1 \).

Wavelet pre-processing of sensor data for symbol sequence generation helps in noise mitigation. Let \( \tilde{f} \) be a noise-corrupted version of the original signal \( f \) expressed as:

\[
\tilde{f} = f + k w,
\]

where \( w \) is additive white gaussian noise with zero mean and unit variance and \( k \) is the noise level. The noise part in Eq. (2.3) would be reduced if the scales over which coefficients are obtained are properly chosen.

For every wavelet, there exists a certain frequency called the center frequency \( F_{c} \) that has the maximum modulus in the Fourier transform of the wavelet. The pseudo-frequency \( f_{p} \) of the wavelet at a particular scale \( \alpha \) is given by the following formula [72]:

\[
f_{p} = \frac{F_{c}}{\alpha \Delta t},
\]

where \( \Delta t \) is the sampling interval. Then the scales can be calculated as follows:

\[
\alpha^{i} = \frac{F_{c}}{f_{p}^{i} \Delta t}
\]

where \( i = 1, 2, ..., \) and \( f_{p}^{i} \) are the frequencies that can be obtained by choosing the locally dominant frequencies in the Fourier transform. The maximum pseudo-frequency \( f_{p}^{\text{max}} \) should not exceed the Nyquist frequency [72]. Therefore, the sampling frequency \( f_{s} \) for acquisition of time series data should be selected at least twice the larger of the maximum pseudo-frequency \( f_{p}^{\text{max}} \) and the signal bandwidth \( B \), i.e., \( f_{s} \geq 2 \max(f_{p}^{\text{max}}, B) \).
2.3 Symbolic Dynamics and Encoding

A crucial step in SDF is partitioning of the data space for symbol sequence generation [22]. Various partitioning techniques have been suggested in literature for symbol generation, which include variance-based [73], entropy-based [74], and hierarchical clustering-based [75] methods. Another partitioning scheme based on symbolic false nearest neighbors (SFNN) was reported in [26]. These techniques rely on partitioning the phase space and may become cumbersome and extremely computation-intensive if the dimension of the phase space is large. Moreover, if the data set is noise-corrupted, then the symbolic false neighbors would rapidly grow in number and require a large symbol alphabet to capture the pertinent information. Therefore, symbolic sequences as representations of the system dynamics should be generated by alternative methods because phase-space partitioning might prove to be a difficult task.

Technical literature has suggested appropriate transformation of the signal before employing the partitioning method for symbol generation [10]. One such technique is the analytic-signal-space partitioning (ASSP) [28] that is based on the analytic signal which provides the additional phase information in the sensor data. The wavelet-transformed space partitioning (WTSP) [27] is well-suited for time-frequency analysis of non-stationary signals, noise attenuation, and reduction of spurious disturbances from the raw time series data without any significant loss of pertinent information [29][24]. Since SFNNP and ASSP may require additional pre-processing of the time series for denoising, this chapter uses WTSP for construction of symbolic representations of sensor data as explained below.

Definition 2.3.1. (Wavelet Surface Profile) Let $\mathcal{H} \triangleq \{(i, j) : i, j \in \mathbb{N}, 1 \leq i \leq m, 1 \leq j \leq n\}$ be the set of coordinates consisting of $m \times n$ pixels denoting the scale-shift data points. Let $\mathcal{R}$ denote the interval that spans the range of wavelet coefficient amplitudes. Then, a wavelet surface profile is defined as

$$\mathcal{S}: \mathcal{H} \rightarrow \mathcal{R}$$

Figure 2.1 shows an illustrative example of transformation of the time series in Fig. 2.1(a) to a (two-dimensional) wavelet image in Fig. 2.1(b). The amplitudes
Figure 2.1. Symbol image generation via wavelet transform of the sensor time series data and partition of the wavelet surface in ordinate direction of the wavelet coefficients over the scale-shift domain are plotted as a surface. Subsequently, symbolization of this wavelet surface leads to the formation of a symbolic image as shown in Fig. 2.1(c), where the $x - y$ coordinates of the wavelet surface profiles denote the shifts and the scales respectively, and the $z$-coordinate (i.e., the surface height) denotes the pixel values of wavelet coefficients.

The wavelet surface profiles are partitioned such that the ordinates between the maximum and minimum of the coefficients along the $z$-axis are divided into regions by different planes parallel to the $x-y$ plane. For example, if the alphabet is chosen as $\Sigma = \{a, b, c, d\}$, i.e., $|\Sigma| = 4$, then three partitioning planes divide the ordinate (i.e., $z$-axis) of the surface profile into four mutually exclusive and exhaustive regions, as shown in Fig. 2.1 (b). These disjoint regions form a partition, where each region is labeled with one symbol from the alphabet $\Sigma$. If the intensity of a pixel is located in a particular region, then it is coded with the symbol associated with that region. As such, a symbol from the alphabet $\Sigma$ is assigned to each pixel corresponding to the region where its intensity falls. Thus, the two-dimensional array of symbols, called symbol image, is generated from the wavelet surface profile, as shown in Fig. 2.1 (c).

Definition 2.3.2. (Symbolization) Given the symbol alphabet $\Sigma$, let the partitioning of the interval $\mathcal{R}$ be defined by a map $P : \mathcal{R} \to \Sigma$. Then, the symbolization of a wavelet surface profile is defined by a map $S_\Sigma \equiv P \circ S$ such that

$$S_\Sigma : \mathcal{H} \to \Sigma$$

that labels each pixel of the image to a symbol in $\Sigma$. 
The surface profiles are partitioned by using either the uniform partitioning (UP) or the maximum entropy partitioning (MEP) methods [10, 27, 28], as illustrated in Figs. 2.2(a) and 2.2(b), respectively. If the partitioning planes are separated by equal-sized intervals, then the partition is called the uniform partitioning (UP). Intuitively, it is more reasonable if the information-rich regions of a data set are partitioned finer and those with sparse information are partitioned coarser. To achieve this objective, the maximum entropy partitioning (MEP) method has been adopted in this dissertation such that the entropy of the generated symbols is maximized. The procedure for selection of the alphabet size $|\Sigma|$, followed by generation of a MEP, has been reported in [24]. In general, the choice of alphabet size depends on specific data set and experiments. The partitioning of wavelet surface profiles to generate symbolic representations enables robust feature extraction, and symbolization also significantly reduces the memory requirements [24].

For the purpose of pattern classification, the reference data set is partitioned with alphabet size $|\Sigma|$ and is subsequently kept constant. In other words, the structure of the partition is fixed at the training stage and this partition serves as the reference frame for subsequent data analysis [10].

2.4 Feature Extraction via Construction of Probabilistic Finite State Automata

This section presents construction of a probabilistic finite state automaton (PFSA) for feature extraction based on the symbol image generated from a wavelet surface profile.
profile (as shown in Fig. 2.1).

### 2.4.1 Conversion from Symbol Image to State Image

For analysis of (one-dimensional) time series, a PFSA is constructed such that its states represent different combinations of blocks of symbols on the symbol sequence. The edges connecting these states represent the transition probabilities between these blocks [10][24]. Therefore, for analysis of (one dimensional) time series, the “states” denote all possible symbol blocks (i.e., words) within a window of certain length. Let us now extend the notion of “states” on a two-dimensional domain for analysis of wavelet surface profiles via construction of a “state image” from a “symbol image”.

**Definition 2.4.1.** (State) Let $\mathcal{W} \subset \mathcal{H}$ be a two-dimensional window of size $(\ell \times \ell)$ that is denoted as $|\mathcal{W}| = \ell^2$. Then, the state of a symbol block formed by the window $\mathcal{W}$ is defined as the configuration $q = S_\Sigma(\mathcal{W})$.

Let the set of all possible states (i.e., two-dimensional words or blocks of symbols) in a window $\mathcal{W} \subset \mathcal{H}$ be denoted as $\mathcal{Q} \triangleq \{q_1, q_2, ..., q_{|\mathcal{Q}|}\}$, where $|\mathcal{Q}|$ is the number of (finitely many) states. Then, $|\mathcal{Q}|$ is bounded above as $|\mathcal{Q}| \leq |\Sigma|^{|\mathcal{W}|}$; the inequality is due to the fact that some of the states might have zero probability of occurrence. Let us denote $\mathcal{W}_{i,j} \subset \mathcal{H}$ to be the window where $(i, j)$ represents the coordinates of the top-left corner pixel of the window. In this notation, $q_{i,j} = S_\Sigma(\mathcal{W}_{i,j})$ denotes the state at pixel $(i, j) \in \mathcal{H}$. Thus, every pixel $(i, j) \in \mathcal{H}$ corresponds to a particular state $q_{i,j} \in \mathcal{Q}$ on the image. Every pixel in the image $\mathcal{H}$ is mapped to a state, excluding the pixels that lie at the periphery depending on the window size. Figure 2.3 shows an illustrative example of the transformation of a symbol image to the state image based on a sliding window $\mathcal{W}$ of size $(2 \times 2)$. This concept of state formation facilitates capturing of long range dynamics (i.e., word to word interactions) on a symbol image.

### 2.4.2 State Compression

In general, a large number of states would require a high computational capability and hence might not be feasible for real-time applications. The number of states,
Figure 2.3. Conversion of the symbol image to the state image

$|Q|$, increases with the window size $|W|$ and the alphabet size $|\Sigma|$. For example, if $\ell = 2$ and $|\Sigma| = 4$, then the total number of states are $|Q| \leq |\Sigma|^\ell = 256$. Therefore, for computational efficiency, it is necessary to compress the state set $Q$ to an effective reduced set $O \triangleq \{o_1, o_2, \ldots, o_{|O|}\}$ [24] that enables mapping of two or more different configurations in a window $W$ to a single state. State compression must preserve sufficient information as needed for pattern classification, albeit possibly lossy coding of the wavelet surface profile.

In view of the above discussion, a probabilistic state compression method is employed, which chooses the $m$ most probable symbols, from each state as a representation of that particular state. In this method, each state consisting of $\ell \times \ell$ symbols is compressed to a reduced state of length $m < \ell^2$ symbols by choosing the top $m$ symbols that have the highest probability of occurrence arranged in descending order. If two symbols have the same probability of occurrence, then either symbol may be preferred with equal probability. This procedure reduces the state set $Q$ to an effective set $O$, where the total number of compressed states is given as: $|O| = |\Sigma|^m$. For example, if $|\Sigma| = 4$, $|W| = 4$ and $m = 2$, then the state compression reduces the total number of states to $|O| = |\Sigma|^m = 16$ instead of 256. This method of state compression is motivated from the renormalization methods in Statistical Physics that are useful in eliminating the irrelevant local information on lattice spin systems while still capturing the long range dynamics [21]. The choice of $|\Sigma|$, $\ell$ and $m$ depends on specific applications and noise level as well as the available computational power, and is made by an appropriate tradeoff between robustness to noise and capability to detect small changes. For example, a large
alphabet may be noise-sensitive while a small alphabet could miss the information of signal dynamics [24].

2.4.3 Construction of PFSA

A probabilistic finite state automaton (PFSA) is constructed such that the states of the PFSA are the elements of the compressed state set \( O \) and the edges are the transition probabilities between these states. Figure 2.4 shows an example of a typical PFSA with four states. The transition probabilities between states are defined as:

\[
P(o_k | o_l) = \frac{N(o_l, o_k)}{\sum_{k'=1,2,...,|O|} N(o_l, o_{k'})} \quad \forall \, o_l, o_k \in O \tag{2.8}
\]

where \( N(o_l, o_k) \) is the total count of events when \( o_k \) occurs adjacent to \( o_l \) in the direction of motion. The calculation of these transition probabilities follows the principle of sliding block code [20]. A transition from the state \( o_l \) to the state \( o_k \) occurs if \( o_k \) lies adjacent to \( o_l \) in the positive direction of motion. Subsequently, the counter moves to the right and to the bottom (row-wise) to cover the entire state image, and the transition probabilities \( \varphi(o_k | o_l) \), \( \forall \, o_l, o_k \in O \) are computed using Eq. (2.9). Therefore, for every state on the state image, all state-to-state transitions are counted, as shown in Fig. 2.5. For example, the dotted box in the bottom-right corner contains three adjacent pairs, implying the transitions \( o_l \rightarrow
Figure 2.5. An example of feature extraction from the state image

\( o_2, o_1 \rightarrow o_3, \) and \( o_1 \rightarrow o_4 \) and the corresponding counter of occurrences \( N(o_1, o_2), N(o_1, o_3) \) and \( N(o_1, o_4) \), respectively, are increased by one. This procedure generates the stochastic state-transition probability matrix of the PFSA given as:

\[
\Pi = \begin{bmatrix}
\varphi(o_1|o_1) & \ldots & \varphi(o_{|O|}|o_1) \\
\vdots & \ddots & \vdots \\
\varphi(o_1|o_{|O|}) & \ldots & \varphi(o_{|O|}|o_{|O|})
\end{bmatrix}
\tag{2.9}
\]

where \( \Pi = [\pi_{jk}] \) with \( \pi_{jk} = \varphi(o_k|o_j) \). Note: \( \pi_{jk} \geq 0 \forall j, k \in \{1, 2, \ldots, |O|\} \) and \( \sum_k \pi_{jk} = 1 \forall j \in \{1, 2, \ldots, |O|\} \).

In order to extract a low-dimensional feature vector, the stationary state probability vector \( p \) is obtained as the left eigenvector corresponding to the (unique) unity eigenvalue of the (irreducible) stochastic transition matrix \( \Pi \). The state probability vectors \( p \) serve as the "feature vectors" and are generated from different data sets from the corresponding state transition matrices. These feature vectors are also denoted as "patterns" in this dissertation.

### 2.4.4 Summary of SDF for feature extraction

The major steps of SDF for feature extraction based on the wavelet surface partitioning method presented in this chapter are summarized below:

- Acquisition of time series data from appropriate sensor(s) and pre-processing the data as necessary;
- Wavelet transform of the time series data with appropriate scales to generate
the wavelet surface profile;

- Partitioning of the wavelet surface profile and generation of the corresponding symbol image;
- Conversion from symbol image to state image via probabilistic state compression strategy;
- Construction of PFSA and computation of the state transition matrices that in turn generate the state probability vectors as the feature vectors.

The advantages of SDF for feature extraction and subsequent pattern classification are summarized below:

- Robustness to measurement noise and spurious signals;
- Adaptability to low-resolution sensing due to the coarse graining in space partitions [10];
- Capability for detection of small deviations because of sensitivity to signal distortion;

It is noted that although wavelet transform provides desirable features such as time-frequency localization and noise mitigation, direct partitioning on time-series data suffices to generate separable features for pattern classification in many applications. In these cases, partitioning becomes the most crucial step in SDF. This topic is discussed in Chapter 4 where the optimization of the partitioning schemes is studied to further improve the performance of SDF as the feature extraction tool.

2.5 Feature Selection and Pattern Classification

Supervised learning algorithms usually consist of two parts, namely, feature extraction and pattern classification. A wavelet-based feature extraction technique based on SDF is introduced in the previous sections, and this section presents the pattern classification methods. In many applications, feature selection is performed between the steps of feature extraction and pattern classification to select a subset of the most representative features, thereby improving the classification performance.
2.5.1 Feature Selection

Feature selection, also known as variable selection, attribute selection or variable subset selection, is a technique of selecting a subset of relevant features for building robust learning models. By removing most irrelevant and redundant features from the data, feature selection helps improve the performance of learning models by [76]:

- Alleviating the effect of the curse of dimensionality;
- Enhancing generalization capability;
- Speeding up learning process;
- Improving model interpretability.

Feature selection algorithms typically consist of two parts: feature ranking and subset selection. Feature ranking ranks the feature by a metric and eliminates all features that do not achieve an adequate score. Subset selection searches the set of possible features for the optimal subset. Subset selection evaluates a subset of features as a group for suitability. In general, subset selection algorithms can be broken into Wrappers and Filters, as shown in Fig. 2.6. Wrappers use a search algorithm to search through the space of possible features and evaluate each subset by running a classifier on the subset. Wrappers can be computationally expensive and have a risk of over fitting to the model. Filters are similar to Wrappers in the search approach, but instead of evaluating against a model, a simpler filter is evaluated. Commonly used filter metrics include class separability, correlation, and mutual information. As shown in Fig. 2.6, both wrapper and filter involves a search algorithm. Typical search algorithms include: exhaustive, best first, genetic algorithm, greedy forward selection, and greedy backward elimination [76].
2.5.2 Pattern Classification

Once the patterns are extracted from the observed sensor time series and a subset of features are selected, the next step is to classify these low-dimensional patterns into different categories based on the particular application. Technical literature abounds in diverse methods of pattern classification, such as divergence measure, \(k\)-nearest neighbor (\(k\)-NN) algorithm [77], support vector machine (SVM) [5], and artificial neural network (ANN) [78]. The main focus of this chapter is to develop the tools of Symbolic Dynamic Filtering (SDF) for feature extraction from wavelet surface profiles generated from sensor time series data. Therefore, the SDF method for feature extraction is used in conjunction with the standard pattern classification algorithms.

Pattern classification using SDF-based features is posed as a two-stage problem, i.e., the training stage and the testing stage. The sensor time series data sets are divided into three groups: i) partition data, ii) training data, and iii) testing data. The partition data set is used to generate partition planes that are used in the training and the testing stages. The training data set is used to generate the training patterns of different classes for the pattern classifier. Multiple sets of training data are obtained from independent experiments for each class in order to provide a good statistical spread of patterns. Subsequently, the class labels of the testing patterns are generated from testing data in the testing stage. The partition data sets may be part of the training data sets, whereas the training data sets and the testing data sets must be mutually exclusive.

Figure 2.7 depicts the flow chart of the proposed algorithm that is constructed based on the theory of SDF. The partition data is wavelet-transformed with appropriate scales to convert the one-dimensional numeric time series data into the wavelet image. The corresponding wavelet surface is analyzed using the maximum entropy principle [27][24] to generate the partition planes that remain invariant for both the training and the testing stage. The scales used in the wavelet transform of the partitioning data also remain invariant during the wavelet transform of the training and the testing data. In the training stage, the wavelet surfaces are generated by transformation of the training data sets corresponding to different classes. These surfaces are symbolized using the partition planes to generate the symbol images. Subsequently, PFSAs are constructed based on the corresponding symbol
images, and the training patterns (i.e., state probability vectors $p$ or state transition matrices $Π$) are extracted from these PFSAs. Feature selection is performed on the training data set to select the subset of the representative and discriminative features. The indices of the selected features are recorded, and the patterns with the selected features are used to train the pattern classifier. Similar to the training stage, the PFSA and the associated pattern is generated for different data sets in the testing stage. The pattern with the selected features is then classified into different classes using pattern classifier, such as SVM, $k$-NN and ANN.

Consider a classification problem of $|C|$ classes, where $C$ is the set of class labels. In the training stage, feature vectors $p_{C_i}^{C_j}, j = 1, 2, ..., n_i$ are generated from the training data sets of class $C_i$, where $n_i$ is the number of samples in class $C_i$. The same procedure is carried out for all other classes. In the testing stage, a testing feature vector $p_{test}$ with unknown class labels is generated using SDF. Two examples of using the pattern classifiers with SDF are provided here. For $k$-NN algorithm, the estimated class label of a testing feature vector $p_{test}$ is equal to the most frequent class among the $k$-nearest training features [77]. For SVM, a separating hyperplane/hypersurface is generated based on training feature vectors $(p_{C_i}^{C_j}, j = 1, 2, ...n_i)$. The estimated class label of the testing feature vector $p_{test}$ depends on which side of the hyperplane/hypersurface the testing feature vector falls [5].
This chapter presents the experimental validation of the wavelet-based feature extraction method proposed in Chapter 2 for behavior recognition in mobile robots. The objective here is to identify the robot type and the behavior (i.e., the motion type) based on the time series data obtained from the pressure sensitive floor. These experiments are inspired from various real-life applications of pattern classification, such as (i) classification of enemy vehicles across the battlefield through analysis of seismic and acoustic time series data; and (ii) classification of human and animal movements through analysis of seismic time series data [37].

3.1 Introduction

Classification of patterns in the behavior of dynamical systems is critical for multi-agent coordination [79]. This issue has become increasingly important with technological advancements in mobile robots and networked robotic systems. For example, a robotic platform is often required to make real-time decisions for target tracking [80]. Such decisions could be made based on the ensemble of information acquired from both mobile and stationary sensors on a distributed network. In such applications, it is necessary to rely on sensor time series, because accurate and computationally tractable modeling of robot dynamics in changing environments may not be feasible solely based on fundamental principles of physics.
Hidden Markov models (HMM) are widely used in motion identification with known domain knowledge. One example is that conventional HMM tools have been adopted in the context of robotic soccer [81], where the states and the structure of HMM are fixed *a priori*, and the state transition and observation probabilities are identified using the Baum-Welch algorithm [45]. Chang and Huang [82] constructed a HMM to analyze human motions by making use of the domain knowledge to extract feature vectors. In contrast, this chapter presents feature extraction and pattern classification for identification of mobile robot types and their motions based on time-series analysis with limited or no requirements of domain knowledge about the robot kinematics. Thus, the proposed pattern classification methods have general applicability and are expected to be robust relative to spurious noise and environmental uncertainties.

Three types of motion (e.g., circular, square and random) for two different types of mobile robots (e.g., Pioneer 2AT and Segway RMP) investigated in this chapter, emulate both structured and unstructured movements of inanimate objects (e.g., different types of vehicles) and animate objects (e.g., human and animals) for pattern analysis from time series data of piezoelectric sensors, installed underneath the laboratory floor, which are representatives of seismic signals. This research topic is inspired from various real-life applications of pattern classification. Examples are: (i) identification of enemy vehicles and their operations in air, sea, and ground battlefields [83], and (ii) detection and classification of human and animal movements across borders [84].

The major contribution of this chapter are: (i) development of a data-driven method for behavior recognition of mobile robots that requires limited or no domain knowledge; and (ii) experimental validation on a networked robotic test-bed by comparison with a commonly used tool of feature extraction.

This chapter is organized in five sections (including the present section). Section 3.2 describes the experimental setup and procedure. Section 3.3 provides a description of the application of different pattern analysis methods for robot and motion classification. The experimental results are presented in Sec. 3.4. A brief summary and recommendation for future research are given in Sec. 3.5.
3.2 Experimental Procedure

The experimental set up consists of a wireless network incorporating mobile robots, robot simulators, and distributed sensors as shown in Fig. 3.1 and Fig. 3.2. A major component of the experimental set up is the pressure sensitive floor that consists of distributed piezoelectric wires installed underneath the floor to serve as arrays of distributed pressure sensors. A coil of piezoelectric wire is placed under a $0.65m \times 0.65m$ square floor tile as shown in Fig. 3.2(a) such that the sensor generates an analog voltage due to pressure applied on it. This voltage is sensed by a Brainstem™ microcontroller using one of its 10-bit A/D channels thereby yielding sensor readings in the range of 0 to 1023. The sampling frequency of the pressure sensing device that captures the dynamics of robot motion is 10 Hz, while the maximum pseudo-frequency $f_p^{max}$ is 4.44 Hz (see Sec. 2.2.2). A total of 144 sensors are placed in a $9 \times 16$ grid to cover the entire laboratory environment as shown in Fig. 3.2(b). The sensors are grouped into four quadrants, each being connected to a stack consisting of 8 networked Brainstem microcontrollers for data acquisition. The microcontrollers are, in turn, connected to two laptop computers running Player [85] server that collects the raw sensor data and distributes to any
Table 3.1. Parameters used for Various Types of Motion

<table>
<thead>
<tr>
<th>Motion Type</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circular</td>
<td>Diameter</td>
<td>4m</td>
</tr>
<tr>
<td>Square</td>
<td>Edge length</td>
<td>3m</td>
</tr>
<tr>
<td>Random</td>
<td>Uniform distribution</td>
<td>x-dir 1 to 7 m</td>
</tr>
<tr>
<td></td>
<td></td>
<td>y-dir 1 to 4 m</td>
</tr>
</tbody>
</table>

client over the wireless network for further processing.

Figure 3.1 shows a pair of Pioneer robots and a Segway RMP that have the following features:

- Pioneer 2AT is a four-wheeled robot that is equipped with a differential drive train system and has an approximate weight of 35 kg;
- Segway RMP is a two-wheeled robot (with inverted pendulum dynamics) that has a zero turn radius and has an approximate weight of 70 kg.

Since Pioneer is lighter than Segway and Pioneer’s load on the floor is more evenly distributed, their statistics are dissimilar. Furthermore, since the kinematics and dynamics of the two types of robots are different, the texture of the respective pressure sensor signals are also different [86].

The objective is to identify the robot type and motion type from the time series data. The Segway RMP and Pioneer 2AT robots are commanded to execute three different motion trajectories, namely, random motion, circular motion and square motion. Table 3.1 lists the parameters for the three types of robot motion. In the presence of uncertainties (e.g., sensor noise and fluctuations in robot motion), a complete solution of the robot type and motion identification problem may not be possible in a deterministic setting because the patterns would not be identical for similar robots behaving similarly. Therefore, the problem is posed in the statistical setting, where a family of patterns is generated from multiple experiments conducted under identical operating conditions. The requirement is to generate a family of patterns for each class of robot behavior that needs to be recognized. Therefore, both Segway RMP and Pioneer 2AT robots were made to execute several cycles of each of the three different types of motion trajectories on the pressure sensitive floor of the laboratory environment. Each member of a family represents the pattern of a single experiment of one robot executing a
particular motion profile. As a robot changes its type of motion from one (e.g., circular) to another (e.g., random), the pattern classification algorithm is capable of detecting this change after a (statistically quasi-stationary) steady state is reached. During the brief transient period, the analysis of pattern classification may not yield accurate results because the resulting time series may not be long enough to extract the features correctly.

Figure 3.3(a) shows an example of the sensor reading when the robot moves over it. The voltage generated by the piezoelectric pressure sensor gradually increases as the robot approaches the sensor, and discharge occurs in the sensor when the robot moves away from the sensor and hence the voltage resumes to be 0. The choice of mother wavelet depends on the shape of the sensor signal; the mother wavelet should match the shape of the sensor signal in order to capture the signature of the signal. Haar wavelet ($db1$), as shown in Fig. 3.3(b), is chosen to be the mother wavelet in this application. The sensor data collected by the $9 \times 16$ grid is stacked sequentially to generate a one-dimensional time series. For each motion trajectory consisting of several cycles, the time series data collected from the pressure sensors was divided into 40 to 50 data sets. The length of each data set is $3.0 \times 10^5$ data points, which corresponds to about three minutes of the experiment time. The data sets are randomly divided into half training and half testing. Among the training data, 10 sets are chosen to serve as the partitioning data sets as well.
3.3 Pattern Analysis for Behavior Recognition

This section provides a description of the application of different pattern analysis methods to time series data of pressure sensors for classification of the robots and their motion types.

For feature extraction using SDF, each data set of a family (or class) is analyzed to generate the corresponding state probability vectors (i.e., patterns). Thus, the patterns \( p_{j}^{C_{i}} \), \( j = 1, 2, ..., n_{i} \), are generated for \( n_{i} \) samples in each class \( C_{i} \) corresponding to robot type and motion. Following the SDF procedure, each time-series data set is analyzed using \( |\Sigma| = 8, \ell = 2 \) and \( m = 1 \). Ensemble mean of pattern vectors for different motion profiles of Segway and Pioneer robots is shown in Fig. 3.4. It can be observed in Fig. 3.4 that the state probability vectors of Segway and Pioneer robots are quite distinct. Following Fig. 2.7, for each motion type, the state probability vectors \( p_{j}^{C_{i}} \) were equally divided into training sets and testing sets.
The efficacy of SDF for feature extraction is evaluated by comparison with Principal Component Analysis (PCA) that is a commonly used tool for feature extraction. The concept of PCA and the procedure of using PCA for feature extraction from time-series data are introduced in Appendix A. The time series data are transformed to the frequency domain for noise mitigation and then the standard PCA method is implemented to identify the eigen-directions of the transformed data and to obtain an orthogonal linear operator that projects the frequency-domain features onto a low-dimensional compressed-feature space. For the purpose of comparison, the dimension of this compressed feature space is chosen to be the same as that of the feature vectors obtained by SDF.

In this chapter, standard pattern classification tools such as support vector machine (SVM), k-NN algorithm, radial basis Neural Network (rbfNN), and multilayer perceptron Neural Network (mlpNN) [6, 5] have been used to identify different classes of feature vectors extracted by SDF and PCA. The pattern classifiers identify the type of the robot and its motion profile, based on the acquired statistical patterns. Since, in this pattern classification problem, there are two robots and each robot has three different types of motion profiles, it is natural to formulate this problem as a two-layer classification problem, where the robot type is identified in the first layer followed by identification of the motion type in the second layer. Thus, the above problem is formulated using a tree-structure classification as shown in Fig. 3.5.

Figure 3.5. Tree structure for pattern identification
3.4 Experimental Results and Discussion

The performance comparison between SDF and PCA that are used in conjunction with different classifiers is presented in Table 3.2. The left part of Table 3.2 shows the results of robot type and robot motion classification using SDF for feature extraction, and the right part shows the corresponding results using PCA for feature extraction. As stated earlier, SVM, $k$-NN, rbfNN, and mlpNN have been used as pattern classifiers in both cases. The polynomial kernel is used in SVM, and a neighbor size of $k = 5$ is used in the $k$-NN classifier. The rbfNN uses one hidden layer and one output layer with a single neuron. Optimal training is obtained with 100 neurons in the hidden layer that uses a radial basis function, while the output layer uses a linear transfer function. The mlpNN utilizes a feed-forward back-propagation network that consists of one hidden layer with 50 neurons and an output layer with a single neuron; the tangent sigmoid function has been used in the hidden layers as a transfer function, while the output layer uses a linear function.

It is noted that since the tree structure is used for pattern classification, the motion recognition results are affected by the robot recognition results. For example, the samples that are incorrectly classified in the robot recognition stage will be incorrectly classified for motion recognition also. However, this particular aspect is application dependent and the tree structure for classification can be redesigned accordingly. The classification results are presented in Table 3.2 that show the accuracy percentage equal to $\left(\frac{\text{# correct classifications}}{\text{# total data sets}} \times 100\right)$. In the left part of Table 3.2, the combination of SDF with all four classifiers yield good accuracy in recognizing the robot type, namely, 100% for Segway and more than 94% for Pioneer. Although not explicitly shown in Table 3.2, but all four classifiers successfully identified the three types of motions of the Pioneer robot with 100% accuracy in the robot motion classification stage. The errors in the motion recognition, as seen in Table 3.2, originate from the robot recognition stage. The success rate in recognizing Segway motion is slightly lower due to the following possible reasons: (i) complicated kinematics of the Segway robot, and (ii) the non-stationarity in the samples due to uncertainties in the laboratory environment (e.g., floor friction). It is expected that this accuracy would further improve if the number of stationary
Table 3.2. Results of Robot and Motion Classification

<table>
<thead>
<tr>
<th>Pattern Classifier</th>
<th>Feature Extraction using SDF</th>
<th>Feature Extraction using PCA</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Robot Recognition</td>
<td>Motion Recognition</td>
<td>Robot Recognition</td>
</tr>
<tr>
<td>SVM</td>
<td>Segway</td>
<td>100% (58/58)</td>
<td>Random</td>
</tr>
<tr>
<td></td>
<td>Pioneer</td>
<td>94% (61/65)</td>
<td>Circular</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Square</td>
</tr>
<tr>
<td>k-NN</td>
<td>Segway</td>
<td>100% (58/58)</td>
<td>Random</td>
</tr>
<tr>
<td></td>
<td>Pioneer</td>
<td>94% (61/65)</td>
<td>Circular</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Square</td>
</tr>
<tr>
<td>rbfNN</td>
<td>Segway</td>
<td>100% (58/58)</td>
<td>Random</td>
</tr>
<tr>
<td></td>
<td>Pioneer</td>
<td>97% (63/65)</td>
<td>Circular</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Square</td>
</tr>
<tr>
<td>mlpNN</td>
<td>Segway</td>
<td>100% (58/58)</td>
<td>Random</td>
</tr>
<tr>
<td></td>
<td>Pioneer</td>
<td>100% (65/65)</td>
<td>Circular</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Square</td>
</tr>
</tbody>
</table>
samples is increased. The right part of Table 3.2 shows that PCA yields slightly worse (but still comparable) results than SDF in robot recognition. However, SDF significantly outperforms PCA in motion recognition, because the feature vectors extracted by PCA lack class separability among different types of motions, which in turn yields poor motion recognition accuracy.

Figure 3.6 exhibits the effects of changing the number of neighbors in the $k$-NN classifier on the performance (i.e., error magnitude) and robustness (i.e., error fluctuation) by comparing the SDF-based and PCA-based features. Only odd values of $k$ are shown in Fig. 3.6 because, when using an even value of $k$, it might be necessary to break a tie in the number of nearest neighbors by selecting a random tiebreaker or by including the nearest neighbor among the tied groups [6]. Figure 3.6 shows that SDF-based classification is consistently superior to PCA-based classification in terms of both performance and robustness. For $k \geq 3$, the error in robot identification using SDF-based features has a decreasing trend and negligible fluctuations as the neighbor size increases. The corresponding errors of motion classification for both Segway and Pioneer are nearly consistent without any noticeable fluctuations. In contrast, the error in robot identification using PCA-based features is relatively large and has an increasing trend as the neighbor size
Table 3.3. Comparison of Computational Complexity of Feature Extraction Methods

| Feature Extraction Method | Training Stage | | Testing Stage | |
|--------------------------|----------------|-----------------|-----------------|
|                          | Execution Time | Memory Requirement | Execution Time | Memory Requirement |
| SDF                      | 5.21 sec       | 65.2 MB          | 5.17 sec        | 64.9 MB            |
| PCA                      | 6.62 sec       | 233.85 MB        | 0.04 sec        | 37.5 MB            |

increases. The corresponding error of motion classification for Pioneer suffers from large fluctuations, while these fluctuations for Segway are much smaller although the error magnitude is very large. These results show that the SDF-based features yield a significantly better separability among different classes as compared to PCA-based features.

The proposed SDF-based method has a computational complexity of $O(N)$ for a given algebraic structure of the PFSA, with a leading constant that is proportional to the number of scales used in the wavelet transform [87]. A comparison of the computational complexity of SDF and PCA is presented in Table 3.3 in terms of execution time and memory requirements for processing each data set. For the data set consisting of $3.0 \times 10^5$ data points, which is about 3.5 minutes of the experimentation time, it takes an average of 5.21 seconds for SDF and 6.62 seconds for PCA to process each data set in the training stage, respectively. The memory requirement is 65.2 MB for SDF, and 233.9 MB for PCA. PCA takes longer execution time and consumes more memory in the training stage because it needs to calculate the covariance matrix using all training data sets. In the testing stage, the execution time and memory requirement for SDF are almost the same as those in the training stage, while the PCA requires less time and memory than those in the training stage. Both feature extraction methods have real-time implementation capability since the execution time in the testing stage is much less than the experiment time spent for collecting each data set. The rationale for SDF taking longer time than PCA in the testing stage is that the SDF-based method involves wavelet transformation and PFSA construction from the two-dimensional wavelet image in both training and testing stages, while the PCA-based method only involves Fourier transform and finding the projection of the testing data set using the projection matrix that is already constructed in the training stage; this is a price paid for the superior performance and robustness achieved in SDF-based
feature extraction (see Fig. 3.6 and Table 3.2). It is anticipated that the PCA-based method will be relatively slower if the raw time-series is (more effectively) de-noised by wavelet transform instead of Fourier transform. In these experiments, the data analysis was performed on a 2.83 GHz Quad Core CPU desktop computer with 8.0 GB of RAM.

3.5 Conclusions and Future Work

This chapter presents the application of the wavelet-based feature extraction method for behavior identification of mobile robots in a networked laboratory environment. The wavelet-based feature extraction method is compared with PCA that is a commonly used feature extraction tool. Several standard pattern classifiers are used in conjunction with the proposed method and PCA to classify the extracted patterns. Experimental results indicates PCA has slightly worse and yet comparable results with the proposed method in robot type recognition, but the proposed method outperforms PCA in motion recognition. The proposed method also has low computational complexity in terms of execution time and memory requirement.

The major aspects of this feature extraction methodology are summarized below along with pertinent conclusions:

1. Recognition of type and motion profiles of the robot system: The proposed feature extraction method demonstrates the capability to generate well-conditioned low-dimensional feature vectors from time series data for pattern identification for recognition of robot types and motion profiles.

2. Suppression of noise and spurious disturbances: By appropriate selection of wavelet basis and scale range, the wavelet-transformed signal is significantly de-noised relative to the time domain signal because the noise is incoherent with the wavelet while the signal is strongly coherent with the wavelet bases. The symbols generated from wavelet coefficients capture the signal characteristics with larger fidelity than those obtained directly from the time domain signal.

3. Reduction of computational load in feature extraction: In the conversion from the symbol image to the state image, a probabilistic compression strategy is
constructed to reduce the number of states while retaining the pertinent information. It has been shown numerically that the compression strategy significantly reduces the computational requirements in feature extraction, even if the number of symbols and the neighborhood radii are large.

Further research directions include: (i) exploration of other wavelet transform techniques for wavelet image generation; and (ii) behavior identification of multiple robots simultaneously operating in a dynamic environment.
Anomaly Detection in Nuclear Power Plants

The recent Fukushima Daiichi nuclear disaster, the largest nuclear disaster since the Chernobyl disaster of 1986, has become the focus of public attention due to its serious consequences and environmental impacts. Public concern is growing about the operational safety of nuclear power plants, especially under adverse conditions. Instrumentation and control (I&C) systems are the nervous system of a nuclear power plant. They monitor all aspects of the plants health and help respond with the care and adjustments needed. Modernization of the I&C systems in the existing nuclear power plants has posed new challenges and opportunities for both industry and regulators. With advanced I&C techniques, the control system is able to recover from unanticipated adverse conditions and faults as well as from emergency situations by altering its operational envelope in real time [88].

Effective fault accommodation in the nuclear power plants relies on the performance of the condition monitoring systems. Tools of sensor-data-driven anomaly detection facilitate condition monitoring of dynamical systems especially if the physics-based models are either inadequate or unavailable. Along this line, symbolic dynamic filtering (SDF) has been reported in literature as a real-time data-driven tool of feature extraction for pattern identification from sensor time series. However, an inherent difficulty for a data-driven tool is that the quality of detection may drastically suffer in the event of sensor degradation. This chapter proposes an anomaly detection algorithm for condition monitoring of nuclear power plants,
where symbolic feature extraction and the associated pattern classification are optimized by appropriate partitioning of (possibly noise-contaminated) sensor time series. In this process, the system anomaly signatures are identified by masking the sensor degradation signatures. The proposed anomaly detection methodology is validated on the International Reactor Innovative & Secure (IRIS) simulator of nuclear power plants, and its performance is evaluated by comparison with that of principal component analysis (PCA).

4.1 Motivation

Condition monitoring and timely detection of incipient faults are critical for operational safety and performance enhancement of nuclear power plants. There are various sources of anomalous behavior (i.e., deviation from the nominal condition) in plant operations, which could be the consequence of a fault in a single component or simultaneous faults in multiple components. Often it is difficult for the plant operator to detect the anomaly and locate the associated anomalous component(s), especially if the anomaly is small and evolve slowly. Upon occurrence of an anomalous event and subsequent pervasion of its effects, the operator could be overwhelmed by the sheer volume of information, generated simultaneously from various sources. Therefore, it would be beneficial to develop an automated condition monitoring system to assist the plant operator to detect the anomalies and isolate the anomalous components [2, 89, 90].

Condition monitoring algorithms are primarily divided into two different categories, namely, model-based and data-driven. Both model-based and data-driven techniques have been reported in literature for condition monitoring of nuclear power plants. Examples of model-based condition monitoring can be found in [91, 92]. Among data-driven tools, neural networks (NN) and principal component analysis (PCA)-based tools [93, 94, 95, 96] are most popular.

Although model-based techniques have their advantages in terms of physical interpretation, their reliability and computational efficiency for condition monitoring often decrease as the system complexity increases. On the other hand, data-driven techniques are expected to remain largely reliable and computationally efficient in spite of increased system complexity if the goal is to monitor the
input-output information from an ensemble of (appropriately calibrated) sensors while considering the entire system as a black-box. However, unless the ensemble of acquired information is systematically handled, data-driven techniques may become computationally intensive and the performance of condition monitoring may deteriorate due to sensor degradation. Furthermore, data-driven techniques would require high volume of training data (e.g., component malfunction data in the present context).

A problem with handling time series data is its volume and the associated computational complexity; therefore, the available information must be appropriately compressed via transformation of high-dimensional data sets into low-dimensional features with minimal loss of class separability. A major challenge in any sensor-data-driven detection tool is to identify the actual anomaly in the system in the presence of sensor degradation (e.g., drift and noise) without succumbing to a large number of false alarms or missed detections. The situation becomes even more critical if the control system uses observations from the degraded sensors as feed-back signals and thereby distorts the control inputs. Traditionally, redundant sensors along with methods based on analytic redundancy have been used for sensor anomaly identification [97, 98].

The chapter proposes an alternative approach to the above problem, where different class labels are assigned to data sets that are generated from different plant health conditions. The same class labels are assigned to similar health conditions and the associated sensor data are subjected to different noise variance at respective sensor degradation levels. To this end, data-driven anomaly detection is posed as a multi-class pattern classification problem, where the tasks of class assignment and the SDF-based feature extraction are optimized in a supervised manner to enhance the classification performance [99]. A major step in SDF-based feature extraction is partitioning of time-series data to generate symbol blocks that are subsequently converted to feature vectors by use of the probabilistic finite state automata (PFSA) concept [10, 57]. In this context, the major contributions of this chapter are delineated below.

- Application of symbolic dynamic filtering (SDF) and probabilistic finite state automata (PFSA) concepts for anomaly detection and condition monitoring in nuclear power plants;
• Optimization of data partitioning in an SDF-based feature-extraction setting;
• Construction of a data-driven anomaly detection method at the component level and evaluation of its robustness to sensor noise;
• Evaluation of the proposed SDF-based anomaly detection algorithm relative to PCA that is a commonly used tool for anomaly detection.

The chapter is organized into six sections and an appendix. Appendix B describes the International Reactor Innovative & Secure (IRIS) test-bed of nuclear power plants on which the problem of plant condition monitoring has been formulated and validated. Section 4.2 poses data-driven anomaly detection as a multi-class pattern classification problem. Section 4.3 presents the partitioning optimization methodology in the framework of SDF-based feature extraction for anomaly detection. Section 4.4 explains how anomaly detection algorithms are constructed. Section 4.5 compares the results of the proposed method with those of a PCA-based method on the IRIS test-bed. Section 4.6 summarizes the chapter and makes major conclusions along with recommendations for future research.

4.2 Data-Driven Anomaly Detection posed as a Multi-Class Pattern Classification Problem

The details of the IRIS test-bed are described in Appendix B. This section presents a case study to validate the anomaly detection methodology, in which the reactor coolant pump (RCP) is chosen to be the location of a component-level degradation and the primary coolant temperature ($T_{HL}$) sensor in the hot leg piping is chosen for anomaly detection in the RCP. Since the plant controller receives feedback signals from the $T_{HL}$ sensor, any degradation in this sensor could pervade through the plant, which will potentially affect the outputs of the remaining components due to the inherent electro-mechanical and control feedback. Component-level anomaly detection under different sensor noise variance is posed as a multi-class classification problem in the sequel.

Component-level anomaly detection in nuclear power plants involves identification of the anomaly type and location & quantification of the anomaly level.
Table 4.1. Anomaly Types in Reactor Coolant Pump (RCP)

<table>
<thead>
<tr>
<th>Anomaly Type</th>
<th>RCP Overspeed $\psi_{RCP}$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>0.00 to 0.30</td>
</tr>
<tr>
<td>Medium</td>
<td>0.30 to 0.60</td>
</tr>
<tr>
<td>High</td>
<td>0.60 to 0.90</td>
</tr>
</tbody>
</table>

Table 4.2. Noise Level and SNR in the $T_{HL}$ Sensor

<table>
<thead>
<tr>
<th>Noise Level</th>
<th>Signal to Noise Ratio Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>30 dB to 15 dB</td>
</tr>
<tr>
<td>Level 2</td>
<td>15 dB to 10 dB</td>
</tr>
<tr>
<td>Level 3</td>
<td>10 dB to 7 dB</td>
</tr>
</tbody>
</table>

Although the model configuration in the IRIS test-bed can be easily extended to simultaneous anomalies in multiple components, this chapter deals with a single component, namely, the reactor coolant pump (RCP), where the task is to detect an anomaly and identify its level for (possibly unknown) sensor noise variance. The RCP overspeed percentage ($\psi_{RCP}$) is chosen as a health parameter. Table 4.1 shows the approximate ranges of $\psi_{RCP}$ under different anomaly levels. Here, the low anomaly level indicate very minimal overspeed in RCP and hence also includes the absolute nominal health condition ($\psi_{RCP} = 0$). Similarly, depending on the standard deviation of the noise in the hot-leg coolant temperature sensor $T_{HL}$, three sensor noise levels are selected. Table 4.2 shows the ranges of signal-to-noise ratio (SNR) for different noise levels of the hot-leg coolant temperature sensor ($T_{HL}$). The SNR in decibels (dB) is defined as

$$SNR_{dB} = 10 \log_{10} \left( \frac{P_{signal}}{P_{noise}} \right) = 20 \log_{10} \left( \frac{A_{signal}}{A_{noise}} \right)$$

(4.1)

where $P_{signal}$ is the power of the signal, $P_{noise}$ is the power of the noise, $A_{signal}$ is the root mean square (RMS) amplitude of the signal, and $A_{noise}$ is the RMS value of the noise. Since the $T_{HL}$ responses are close to sine waves (as shown in Fig. 4.3), $A_{signal} \approx \frac{A_{pp}}{2\sqrt{2}}$, where $A_{pp}$ is the peak to peak amplitude and $A_{noise} = \sigma_{T_{HL}}$.

In the above context, $(3 \times 3) = 9$ classes of data sets are chosen to define each class by an RCP anomaly level and a noise level of the $T_{HL}$ sensor. One hundred simulation runs are performed on the IRIS system for each class to generate time series data, among which 50 samples were chosen for training and the remaining
samples for testing. The parameters, RCP overspeed percentage $\psi_{RCP}$ and standard deviation of $T_{HL}$ are chosen randomly from independent uniform distribution such that all of the parameter values are within the prescribed ranges given in Tables 4.1 and 4.2. Figure 4.1(a) plots the generated samples in the two-dimensional parameter space, where different classes of samples are shown in different colors and are marked within the class number. Note that the standard deviation ($\sigma_{T_{HL}}$) in Fig. 4.1(a) shows values of actual standard deviation, not as percents.

Often component degradations in nuclear plants occur on a slow-time scale, i.e., the anomaly gradually evolves over a relatively long time span. Sudden failures in plant components are disastrous and may occur only if a component accumulates sufficient amount of anomaly over a prolonged period and reaches the critical point,
which justify early detection of an anomaly for enhancement of plant operational safety. However, component anomaly at the early stages of a malfunction is usually very small and is difficult to detect, especially under steady-state operations. One way to “amplify” the anomaly signature is to perturb the plant by (quasi-) periodically perturbing the turbine load such that the plant is in a transient state for a period that is sufficiently long for anomaly detection and insignificant from the perspectives of plant operation.

For each sample point in the parameter space, a time series is collected for $T_{HL}$ sensor under persistent excitation of turbine load inputs that have truncated triangular profiles with the mean value of 99% of nominal output power, fluctuations within $\pm$1% and frequency of 0.00125Hz (period $T = 800$ sec). In other words, the turbine load is fluctuating between nominal power (i.e., 335 MW) and 98% of nominal power (i.e., 328.3 MW). Figure 4.2 shows the profile of turbine load and a typical example of turbine output power as a result of fluctuation in turbine load.

For each experiment, the sampling frequency for data collection is 1 Hz (i.e., the inter-sample time of 1 sec) and the length of the perturbation time window
is 2,400 seconds, which generate 2,400 data points. The total perturbation period for each experiment is 3,000 seconds. In this chapter, to eliminate the possible transients, the data for the first 600 seconds have not been used so that quasi-stationarity is achieved. Figure 4.3 shows representative examples of $T_{HL}$ time series data from each of nine classes. Reduction of the perturbation period needs to be investigated for in-plant operations, which is a topic of future research.

With the objective of building a data-driven diagnostic algorithm that is robust to sensor noise (within an allowable range), a data class is defined to be only dependent on the RCP overspeed parameters. Thus, the 9 original classes are reduced to 3 classes as shown in Fig. 4.1(b). This is the final class assignment for the data set, where each class has $(50 \times 3) = 150$ training samples and $(50 \times 3) = 150$ test samples. Thus, the problem of component level anomaly detection is formulated in presence of sensor noise as a multi-class classification problem; in the present scenario, number of classes is 3.

### 4.3 Optimization of Partitioning

Properties and variations of transformation from the symbol space to the feature space have been extensively studied in the Mathematics, Computer Science and especially Data Mining literature. Apparently, similar efforts have not been expended to investigate partitioning of time series data to optimally generate symbol blocks for pattern classification and anomaly detection. Symbolic false nearest neighbor partitioning (SFNNP) [26] optimizes a generating partition by avoiding topological degeneracy. However, a major shortcoming of SFNNP is that it may become extremely computationally intensive if the dimension of the phase space of the underlying dynamical system is large. Furthermore, if the time series data become noise-corrupted, the states of SFNNP rapidly grow in number and thus the partitioning may erroneously require a large number of symbols to capture pertinent information on the system dynamics [28]. This shortcoming could be largely alleviated by wavelet-transformed space partitioning (WTSP) that is particularly effective for noisy data for large-dimensional dynamical systems [10]; maximum entropy partitioning (MEP) was used by Rajagopalan and Ray [27] to generate symbol blocks from time series data by WTSP. Although WTSP is significantly com-
putationally faster than SFNNP and is suitable for real-time applications, WTSP also has several shortcomings such as, requirements of good understanding of signal characteristics for selection of the wavelet basis, identification of appropriate scales, and lossy & non-unique conversion of the two-dimensional scale-shift domain into a single dimension. Subbu and Ray [28] introduced Hilbert-transform-based analytic signal space partitioning (ASSP) as an alternative to WTSP, and Sarkar et al. [58] generalized ASSP for symbolic analysis of noisy signals.

The data partitioning techniques, described above, primarily provide a symbolic representation of the underlying dynamical system under a given quasi-stationary condition, rather than capturing the data-evolution characteristics as a consequence of an anomaly. Next we present a partitioning optimization methodology that endeavors to overcome this shortcoming to make SDF a robust data-driven feature extraction tool for pattern classification and anomaly detection [100, 101, 102].

Optimization criteria that are reported for feature extraction in multi-class classification problems are broadly classified as:

1. Filtering method that makes use of the information content feedback (e.g., Fisher criteria, statistical dependence, and information-theoretic measures) as optimization criteria for feature extraction.

2. Wrapper method that includes the classifier inside the optimization loop to maximize the predictive accuracy (e.g. classification rate using statistical re-sampling or cross-validation [5]).

The wrapper method is adopted here to minimize the classification error on the training set. The rationale for this choice is the non-binary nature of the problem at hand and the possible non-Gaussian distribution of training samples in the feature space.

In a multi-class problem, ideally one should jointly minimize all the off-diagonal elements of the confusion matrix, while maximizing the diagonal elements. However, in that case, the dimension of the objective space blows up with increase in the number of classes, which is obviously impractical. Therefore, two cost functionals may be defined on the confusion matrix by using another penalty weighting matrix, elements of which denote the relative penalty values for different confusions in the classification process [103]. Formally, let there be $C_{l_1}, \cdots, C_{l_n}$ classes
of labeled time-series data given as the training set. A partitioning \( B \) is employed to extract features from each sample and a \( k \)-NN classifier \( K \) is used to classify them. After the classification process, the confusion matrix [104] \( C \) is obtained, where the value of its element \( c_{ij} \) denotes the frequency of data from class \( Cl_i \) being classified as data from class \( Cl_j \). Let \( W \) be the weighting matrix, where the value of its element \( w_{ij} \) denotes the penalty incurred by the classification process for classifying a data set from class \( Cl_i \) as a data set from class \( Cl_j \). With these definitions, the cost due to expected classification error, \( Cost_E \) is defined as:

\[
Cost_E \triangleq \frac{1}{N_s} \left( \sum_i \sum_j w_{ij}c_{ij} \right)
\]

(4.2)

where \( N_s \) is the total number training samples including all classes. The outer sum in the above equation sums the total penalty values for misclassifying each class \( Cl_i \). Thus \( Cost_E \) is related to the expected classification error. Although, in the current formulation, the total penalty values are equally weighted for all classes, that can be changed based on prior knowledge about the data and the user requirements.

It is implicitly assumed in many supervised learning algorithms that the training data set is a statistically similar representation of the whole data set. However, this assumption may not be very accurate in practice. A natural solution to this problem is to choose a feature extractor that minimizes the worst-case classification error [105] as well. In the present setting, that cost due to worst-case classification error, \( Cost_W \) can be defined as:

\[
Cost_W \triangleq \max_i \left( \frac{1}{N_i} \sum_j w_{ij}c_{ij} \right)
\]

(4.3)

where \( N_i \) is the number of training samples in class \( Cl_i \). With such construction, the dimension of the objective space is not a function of the number of classes, which makes it convenient for classification with large number classes. As described earlier, classification needs to be performed on the training data set to calculate the costs during optimization of the feature extractor, i.e., the partitioning.

Figure 4.4 depicts the general outline of the classification process. Labeled time series data from the training set are partitioned and the generated low-dimensional
Feature Set
Costs related to classification error
Tuning of Classifier
Partitioning
Search for SDF
Labeled Time-series Data
Classification Parameters
Optimal Partition
SDF
Feature Set
Classifer
Unlabeled Time-series Data
Test Results
Training Data

Figure 4.4. General framework for optimization of feature extraction

feature vectors (via symbolization and PFSA construction) are fed to the classifier. After classification, the two training error costs defined as above are computed and fed back to the feature extraction block. During classification, the classifier may be tuned to obtain better classification rates. For example, for $k$-NN classifiers [77], choice of neighbor size or the distance metric can be tuned. The partitioning is updated to reduce the cost based on the feedback. The iteration is continued until the set of optimal partitioning in a multi-objective scenario and the correspondingly tuned classifier are obtained. Choice of the optimal partitioning is done using the Neyman-Pearson criterion [106] as described later. After the choice is made, the optimal partitioning and the tuned classifier are used to classify the test data set. Although this is the general framework that is being proposed for the optimization methodology, tuning of the classifier has not been performed in this chapter as the main focus here is to choose the optimal partitioning to minimize the classification error related cost.

Similar to the classical partitioning cases, the Euclidean distance is chosen as the metric. For partitioning optimization, at first, the number of cells $|\Sigma|$ of the
partitioning $\mathcal{B}$ is chosen to be 4 in this case. For computation purpose, a suitably fine grid size depending on the data characteristics is then assumed. It should be clear that each of the grid boundaries denote a possible position of a partitioning cell boundary. In this chapter, the data space region $\Omega$ is divided into 40 grid cells, i.e., 39 grid boundaries excluding the boundaries of $\Omega$ and the alphabet size is selected to be $|\Sigma| = 4$. That is, there are 4 partitioning cells and 3 partitioning boundaries to choose. Hence, the number of elements (i.e., 4-dimensional partitioning vectors) in the space $\mathcal{P}$ of all possible partitioning is: $39 \times 3 = 9139$. Since the cardinality of $\mathcal{P}$ is computationally tractable in this example, a direct search-based Pareto optimization procedure is followed in this chapter. If the partitioning space $\mathcal{P}$ happens to be significantly large such that usage of a direct search approach becomes infeasible for evaluation of all possible partitioning, then other searching schemes (e.g., those based on genetic algorithms [107]) should be applied.

By searching the partition space $\mathcal{P}$, the positions of its elements (i.e., the partitioning vectors) are located in the (two-dimensional) $Cost_E$-$Cost_W$ objective space. The resulting Pareto front is generated by identifying the non-dominated points [106] in the objective space. In the present case, a non-dominated point (i.e., a partitioning vector) is such that no other partitioning has lower values of both $Cost_E$ and $Cost_W$. Finally, the Neyman-Pearson criterion [106] is applied to choose the optimal partitioning $\mathcal{B}^*$ for minimization of the cost $Cost_E$, while not allowing the cost $Cost_W$ to exceed the allowable worst-case classification error. In other words, the optimal partitioning $\mathcal{B}^*$ is the solution to the following constrained optimization problem:

$$\mathcal{B}^* = \arg \min_{\mathcal{B}} Cost_E(\mathcal{B}) \text{ such that } Cost_W(\mathcal{B}) \leq \epsilon$$

(4.4)

where $\epsilon$ is the allowable worst-case classification error that is denoted by the (positive scalar) threshold determined by the user.

### 4.4 Anomaly Detection Algorithm

This section describes how the anomaly detection algorithm is formulated based on the concept of symbolic dynamic filtering (SDF). For comparative evaluation,
another anomaly detection algorithm is constructed based on the principal component analysis (PCA), also known as proper orthogonal decomposition or Karhunen-Loève transformation [7, 6, 5], which is a linear feature extraction technique and computes the largest eigenvectors of the covariance matrix.

A flowchart of the anomaly detection algorithms is shown in Fig. 4.5 and the following steps summarize the procedure.

4.4.1 Symbolic Dynamic Filtering for Anomaly Detection

The following steps summarize the procedure of SDF for anomaly detection.

- **Time series data acquisition** on the fast scale from sensors and/or analytical measurements (i.e., outputs of a physics-based or an empirical model). Data sets of different plant and sensor conditions are collected at slow time epochs, and divided into partitioning, training and testing sets.

- **Partitioning the time series data.** Each segment of the partitioning is assigned a particular symbol from the symbol alphabet set $\Sigma$. This step enables transformation of time series data from the continuous domain to the symbolic domain [20]. The partitioning is generated from the partitioning data and remains fixed for subsequent training and testing stage.

- **Construction of probabilistic finite state automata** from each data set with alphabet size $|\Sigma|$ and window length $D$. Choices of $|\Sigma|$ and $D$ depend on the characteristics of the data. In this chapter, $D = 1$ and $|\Sigma| = 4$ are taken.
• **Generation of state probability vectors.** The state probability vectors \{p\} are recursively computed as an approximation of the natural invariant density of the dynamical system. For each data set, a state probabilistic vector is generated and serve as the feature vector in the subsequent pattern classification tasks.

• **Identification of the class information** The low-dimensional state probability vectors are input to pattern classifiers to identify the class labels. The classifier is trained in a supervised manner with the training patterns. The \(k\)-nearest neighbors (\(k\)-NN) algorithm [6, 5] has been used in this chapter as the pattern classifier due to its simplicity.

### 4.4.2 Principal Component Analysis for Anomaly Detection

The details of PCA algorithm for anomaly detection are introduced in Appendix A. By using PCA as the feature extraction tool, low-dimensional representation of the original data matrix is obtained. The next step is to use a pattern classifier to determine the class labels of the testing data sets. For performance comparison of SDF with PCA, the \(k\)-NN algorithm is also used as the pattern classifier for PCA. Rows of the \(M \times m\) matrices \(Y_{\text{train}}\) and \(Y_{\text{test}}\) serve as the training patterns and testing patterns, respectively. Other common pattern classifiers, namely, linear discriminant analysis (LDA) and least squares algorithm (LS) [6][5] have also been used for comparative evaluation, as reported later in the chapter.

### 4.5 Results and Discussion

This section presents pertinent results for the case study of anomaly detection in the reactor coolant pump (RCP) for comparative evaluation of the optimal partitioning-based SDF tool with those based on classical methods of partitioning as well as PCA.

At the beginning of the optimization procedure, a weighting matrix \(W\) needs to be defined to calculate the cost functionals \(Cost_E\) and \(Cost_W\) from the confusion
matrix for the training data set. In this case study, $W$ is defined according to the adjacency properties of classes in the parameter space, i.e., $w_{ii} = 0 \forall i \in \{1, 2, 3\}$, i.e., there is no penalty for correct classification. The weights are selected as: $w_{ij} = |i - j|$, $\forall i \in \{1, 2, 3\}$, i.e., given that a data sample originally from $Cl_i$ is classified as a member of $Cl_j$, the penalty incurred by the classification process increases with increase in the separation of $Cl_i$ and $Cl_j$ in the parameter space. Then, it follows that:

$$W = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$$

The data space region $\Omega$ is divided into 40 grid cells, i.e., 39 grid boundaries excluding the boundaries of $\Omega$. Each partitioning in the space $\mathcal{P}$ is evaluated by calculating $Cost_E$ and $Cost_W$ to identify the optimal partitioning. Figure 4.6 shows a relevant region of the (two-dimensional) $Cost_E$-$Cost_W$ objective space, where the elements of the space $\mathcal{P}$ are located. The Pareto front is also generated from this evaluation. The threshold $\epsilon$, i.e., the maximum allowable $Cost_W$ is taken to be 0.041 in this case and the optimal partitioning (OptP) is chosen by the Neyman-Pearson criterion as discussed earlier. Thus, the user has the option of choosing a classification operating point with selected values of $Cost_E$ and $Cost_W$. 

**Figure 4.6.** Two dimensional objective space for partitioning optimization
Figure 4.7. Feature space of the training set using optimal partitioning

Figure 4.8. Feature space of training set – uniform partitioning (UP)

Figure 4.9. Feature space of training set – maximum entropy partitioning (MEP)
The Pareto front [106] is generated after the threshold \( \epsilon \) is chosen. Locations of the classical partitioning (i.e., uniform (UP) and maximum entropy (MEP)) are also plotted along with the elements of \( P \) in the figures for comparative evaluation.

For SDF analysis, the alphabet size is taken to be \( |\Sigma| = 4 \) and the depth for constructing PFSA is taken to be \( D = 1 \). Features are classified by a \( k \)-NN classifier using the Euclidean distance metric. Figure 4.7 shows locations of the training features in the three-dimensional plot using first three linearly independent elements of the feature vectors obtained by using the chosen optimal partitioning, OptP. Note, only \((|\Sigma| - 1)\) out of its \(|\Sigma|\) elements of a feature vector are linearly independent, because a training feature vector, \( \mathbf{p} \) is also a probability vector, i.e., the sum of its elements is constrained to be equal to 1. Levels of RCP anomalies are represented in different symbols: blue • indicates low anomaly, red + indicates medium anomaly, and black × indicates high anomaly. The class separability is retained by the feature extraction (partitioning) process even after compressing a time series (with 2,400 data points) into 3 numbers.

For comparison purpose, classical partitioning schemes, such as, Uniform Partitioning (UP) and Maximum Entropy Partitioning (MEP) are also used with the same alphabet size, \( |\Sigma| = 4 \). Figures 4.8 and 4.9 show the location of each training time series in the three dimensional (using first three linearly independent elements of the feature vectors) feature space plot using UP and MEP, respectively. It is observed that the class separation in Fig. 4.8 is not as good as that in Fig. 4.7, especially the separation between low/medium anomaly samples and high anomaly samples. The samples of the same class in Fig. 4.9 do not cluster as close as that in Fig. 4.7; some of the low anomaly samples (i.e., those with blue dots below the red + cluster in Fig. 4.9) are closer to the medium anomaly cluster than their own cluster.

Finally, the confusion matrices [104] for the SDF-based methods (OptP, UP and MEP) with \( k \)-NN on the test data set are given by \( C_{test}^{OptP+kNN} \), \( C_{test}^{UP+kNN} \) and \( C_{test}^{MEP+kNN} \), respectively.

\[
C_{test}^{OptP+kNN} = \begin{pmatrix}
142 & 8 & 0 \\
14 & 136 & 0 \\
0 & 4 & 146
\end{pmatrix}
\]
In a confusion matrix, its elements $c_{ij}$ denote the frequency of data in the class $C_{i}$ being classified as data in class $C_{j}$. The diagonal elements $c_{ii}$ represent the frequency of the data that are correctly classified. If all data are classified correctly, then all the off-diagonal elements of the confusion matrix must be zeros. There are fewer off-diagonal elements in $C_{test}^{OptP+kNN}$, which implies that the expected classification error of using OptP is smaller than those using MEP and UP. The minimum diagonal element in $C_{test}^{OptP+kNN}$ is larger than those in $C_{test}^{UP+kNN}$ and $C_{test}^{MEP+kNN}$, which implies that the worst case error of using OptP is smaller than those using MEP and UP.

For comparative evaluation, the data sets are analyzed using other common pattern recognition tools [6][5]. In this case study, principal component analysis (PCA) is used as the feature extraction tool, while $k$-NN, linear discriminant analysis (LDA) and least squares algorithm (LS) are used as the pattern classifiers. For PCA anomaly detection, $M = 450$, $N = 2400$, and by choosing $\eta = 0.90$, the corresponding number of largest eigenvalues turns out to be $m = 20$, as explained in Appendix A. The confusion matrices for PCA with $k$-NN, LDA and LS are given by $C_{test}^{PCA+kNN}$, $C_{test}^{PCA+LDA}$ and $C_{test}^{PCA+LS}$, respectively. An inspection of the confusion matrices reveals that PCA-based anomaly detection algorithm yields similar performance in identifying the low and high level anomalies as the SDF-based algorithm, but the performance in identifying the medium level anomaly is significantly worse.

$$
C_{test}^{UP+kNN} = \begin{pmatrix}
145 & 5 & 0 \\
17 & 132 & 1 \\
0 & 6 & 144
\end{pmatrix}
$$

$$
C_{test}^{MEP+kNN} = \begin{pmatrix}
143 & 7 & 0 \\
18 & 132 & 0 \\
0 & 5 & 145
\end{pmatrix}
$$

$$
C_{test}^{PCA+kNN} = \begin{pmatrix}
149 & 1 & 0 \\
40 & 110 & 0 \\
0 & 4 & 146
\end{pmatrix}
$$
Figure 4.10. Classification error vs. neighbor size in $k$-NN classifiers

$$
C_{test}^{PCA+LDA} = \begin{pmatrix}
136 & 14 & 0 \\
18 & 126 & 6 \\
0 & 3 & 147 \\
\end{pmatrix}
$$

$$
C_{test}^{PCA+LS} = \begin{pmatrix}
145 & 5 & 0 \\
27 & 115 & 8 \\
0 & 0 & 150 \\
\end{pmatrix}
$$

Figure 4.10 shows how the neighbor size in the $k$-NN classifiers affect the classification performance, and the performances of OptP, MEP, UP and PCA with different neighbor size $k$ are compared. Only odd values of neighbor size are shown because when using an even value for $k$, it might be necessary to break a tie in the number of nearest neighbors by selecting a random tiebreaker or using the nearest neighbor among the tied groups. It is seen in Fig. 4.10 the classification performance of SDF-based methods is consistently better than that of PCA-based method and almost independent of the neighbor size, whereas the classification error of PCA increases with the number of neighbor size. This results show that the feature vectors extracted by SDF retain better separability among classes than those extracted by PCA.

Table 4.3 presents the comparison of the classification error related costs for SDF-based methods and PCA-based methods on the test data set. The observation made from these results indicate that the classification performance of SDF-based methods are superior to that of the PCA-based methods. The classification per-
Table 4.3. Comparison of Classification Performances of Different Methods on Testing Data Set (50 × 9 Samples)

<table>
<thead>
<tr>
<th>Methods</th>
<th>CostE</th>
<th>CostW</th>
<th>Classification Error %</th>
</tr>
</thead>
<tbody>
<tr>
<td>OptP+kNN</td>
<td>0.0578</td>
<td>0.0933</td>
<td>5.78</td>
</tr>
<tr>
<td>UP+kNN</td>
<td>0.0644</td>
<td>0.1200</td>
<td>6.44</td>
</tr>
<tr>
<td>MEP+kNN</td>
<td>0.0667</td>
<td>0.1200</td>
<td>6.67</td>
</tr>
<tr>
<td>PCA+kNN</td>
<td>0.1000</td>
<td>0.2667</td>
<td>10.00</td>
</tr>
<tr>
<td>PCA+LDA</td>
<td>0.0911</td>
<td>0.1200</td>
<td>9.11</td>
</tr>
<tr>
<td>PCA+LS</td>
<td>0.0889</td>
<td>0.1800</td>
<td>8.89</td>
</tr>
</tbody>
</table>

formance of MEP and UP are close, and UP performs slightly better in this case study. The costs due to worst-case classification error and expected classification error are reduced compared to that of the uniform partitioning scheme by optimizing the partitioning process over a representative training set.

It is noted that, for some problems, the classical partitioning schemes may perform similar to the optimal one. Therefore, the optimization procedure may also be used to evaluate the capability of any partitioning scheme towards achieving a better classification rate. The evaluation can be performed by using a part of the labeled training data set as the validation set. Although the construction of the cost functions allow solutions of problems with a large number of classes, its upper limit could be constrained by the alphabet size used for data partitioning that determines the dimension of the feature space.

4.6 Conclusions and Future Work

This chapter presents a data-driven method for detection of component-level anomalies in nuclear power plants with sensors that are subjected to different noise variance. The task of anomaly detection is viewed as a multi-class pattern classification problem, where feature extraction is optimized to enhance the classification rate. The proposed methodology efficiently (i.e., with low memory requirements and fast execution) compresses a high-volume database to execute anomaly detection for a large number of classes. It has been shown that the SDF-based anomaly detection algorithm has superior performance compared with the PCA-based anomaly detection algorithm.
This proposed method is tested on the IRIS simulator for the white Gaussian noise that is a common representation of sensor uncertainties. If exact specifications of the sensor noise are available, the anomaly detection algorithm should be trained based on these specifications. In the absence of this information, the best choice would be to use white Gaussian noise at different levels of signal-to-noise ratio (SNR), which is done in this chapter.

Scalability is a critical issue for data-driven supervised methods of anomaly detection. A natural way to circumvent this problem is to perform anomaly isolation before estimating a particular anomaly level. Furthermore, although this chapter has shown application of the developed feature extraction method to component anomaly detection problems, the technique is general enough to be used for detecting sensor and actuator anomalies under different architectures.

Potential applications of the proposed anomaly detection method include real-time detection of component-level anomalies in nuclear power plants with low false alarm and missed detection rates. While the training stage could be accomplished in a nuclear power plant simulator with high fidelity (e.g., scaled experimental reactor [108] or computer simulator [109]), the testing stage should be ideally performed in real time in an operating nuclear power plant. However, based on the simulation results only, it is found that the plant perturbation needed to execute the proposed anomaly detection method require small induced changes in the turbine load (e.g., less than 2% nominal load) over a short period, which does not have any significant adverse effects on stability and performance of the plant operation.

While there are research issues in this field [21] that are require in-depth research, the following areas are recommended for future research:

- Usage of other classifiers (e.g., Support Vector Machines (SVM) and Gaussian Mixture Models (GMM)) for comparison with the proposed method’s performance;
- Inclusion of the step of tuning the classifier inside the optimization loop as described in the general framework shown in Fig. 4.4;
- Extension of the proposed anomaly detection method to accommodate other types of sensor degradation (e.g., bias and drifting);
- Investigation of anomalous situations with simultaneous anomalies in multiple plant components, actuators, and sensors;
• Reduction of the plant perturbation period for the testing phase without any significant compromise of detection performance.
Chapter 5

Multi-Modal Sensor Fusion for Personnel Detection

This chapter presents a multi-level data fusion method based on unattended ground sensors (UGS) that are widely used to monitor human activities, such as pedestrian motion and detection of intruders in a secure region. The UGS systems are usually equipped with multi-modal sensing devices to enhance the performance of personnel detection. An information fusion technique at the data level, called the cross machine, has been adopted in this chapter to model relational dependencies among different sensing modalities, and the features obtained from the cross machine are combined to generate the patterns. The proposed multi-level fusion method has been validated on data sets of seismic, passive infrared (PIR), and acoustic sensors for target classification. The results show that the proposed sensor data fusion yields superior performance in terms of target classification relative to what would be achieved by isolated analysis of these data sets.

5.1 Introduction

The surveillance of U.S. borders is a crucial challenge to homeland security. To this end, infeasible manned approaches are being replaced by unmanned sensors and vehicles, supported by new information processing technologies. For example, unattended ground sensors (UGS) have been widely used to monitor human activities, such as pedestrian motion and detection of intruders in a secure region.
Such UGS are usually lightweight devices that automatically monitor the local activities in-situ, and transfer target detection and classification reports to some higher level processing center. Commercially available UGS systems make use of multiple sensing modalities (e.g., acoustic, seismic, passive infrared, magnetic, electrostatic, and video). However, these systems have their own shortcomings for data-driven situation assessment and heterogeneous sensor data fusion. Previous efforts in this area have encountered difficulties due to intolerably high rates of data transfer and misclassification; in particular, high false-alarm rates have resulted from inadequate onboard processing algorithms [110]. Furthermore, limited battery operating life have made power consumption a critical concern for both sensing and information communication [111].

In a personnel detection problem, the targets usually include human, vehicles, and animals. Discriminating vehicle from other targets is a relatively less challenging problem, because the loud and periodic signatures emitted by the vehicle engine are easy to be identified. While small animals (e.g., rabbits, beavers, and squirrels) may not generate sufficient signal strength to trigger the sensors, they are often triggered by medium to large animals (e.g., deer, burro, moose, and livestock). Discriminating noise-contaminated signals of human footsteps from those of animals is a challenging task because signal-to-noise ratio (SNR) of footsteps decreases rapidly with the distance between the sensor and the target (e.g., pedestrian and animal). Furthermore, the footprint signals may vary differently for various types of targets in different environments [112].

Seismic, PIR and acoustic sensors are the most common sensing modalities in personnel detection. Seismic sensors are widely used because they are relatively less sensitive to Doppler effects environment variations as compared to acoustic sensors [113]. Current personnel detection methods using seismic signals can be classified into three categories, namely, time domain methods [114], frequency domain methods [115], and time-frequency domain methods [113, 116]. Recent research has relied on time-frequency domain methods, such as wavelet transform-based methods [117]. Passive infrared (PIR) sensors are widely used for motion detection, and are well-suited for UGS systems due to low power consumption. PIR sensors have been reported for moving targets detection and localization [118]; however, similar effort for target classification has not been reported in open liter-
ature, although PIR sensor signals also contain discriminative information in the
time-frequency domain. Acoustic sensors are widely used in vehicle detection [119]
because the signals emitted by the vehicle engines contain discriminative infor-
mation which can be represented by cepstral features, spectral features, and time
domain features. Major issues of using acoustic sensors in personnel detection are
vulnerability against wind noises and non-stationary signal behavior of moving
targets.

Collaborative target detection and classification using multimodal sensor fu-
sion would increase the overall performance because the heterogeneous sensors can
complement each other. This chapter introduces a multi-level fusion method that
combines a data-level fusion technique called cross machine to model the relational
dependencies across heterogeneous sensors at the data level and a feature-level fu-
sion using class separability measure. Cascaded filter structure is used to first
select the best-performing relational patterns and then select a subset of features
with reliable class discriminative information. The proposed method is validated
on a dataset collected from the border area using seismic, PIR and acoustic sen-
sors. Performance of the multi-level fusion is compared with the ones obtained
from single-modal sensors, data-level fusion and decision-level fusion.

The remainder of this chapter is organized as follows: Section 5.2 describe the
architecture of traditional and the proposed sensor fusion method. Section 5.3
presents the details of the cross machine for data fusion, while Sec. 5.4 discusses
feature selection method for feature fusion. The results of field data analysis is
presented in 5.5, and Sec. 5.6 concludes this chapter.

5.2 The Architecture of Multi-Level Fusion

Information fusion can be implemented at different levels: data level, feature level,
and decision level. A review of fusion at different levels is given in Appendix C.
Kalman filter is widely used for data fusion; feature selection is usually performed
to combine the features from different sensors for feature fusion; Dempster-Shafer
evidence theory and Bayesian network are widely used for decision fusion [120][121].
Data fusion has the least information loss, but it may be computationally expen-
sive and vulnerable to sensor degradation. Decision fusion is robust against small
disturbances in the signal, but it may suffer from information loss since the fusion is performed at a high level and much of the complementary details among different sensors might have been lost. These concerns are alleviated by multi-level fusion in which different levels of fusion schemes are integrated to reduce the information loss and improve the classification performance. The concept of multi-level fusion has been applied to various areas, including land mine detection [122], automotive safety [123], missile defense [124], fire detection [125], structural health monitoring [126], and robotics [127].

Figure 5.1 shows the comparison between traditional fusion methods and the proposed multi-level fusion method. In traditional methods, the information from sensors are integrated either at data level, feature level or decision level. In multi-level fusion method, data fusion is performed between each sensor pairs to model the relational dependencies at data level. A technique called “cross machine” [59, 60] is used and the details of the technique is presented in Sec. 5.3. Relational patterns are generated from the cross machines and combined in the feature fusion step that is introduced in Sec. 5.4. Feature selection is implemented in this step to choose the features that contain discriminative information. The selected features are identified by pattern classifier to determine the class type.
5.3 Data Fusion - Cross Machine

This section presents the concept of cross machine that models the relational dependencies across different sensing modalities at the data level. The concept of cross machine was first introduced by Srivastav et al. [59] for situation assessment in sensor networks and followed by Sarkar et al. [60] for fault diagnosis in aircraft gas turbine engines. In [60], time-series data are generated by the sensors that monitor the health condition of the aircraft gas turbine engine and discretized into symbol sequences. Cross machines are constructed to capture the relational dependencies among the symbol sequences and generate feature vectors for fault diagnosis.

In the personnel detection problem, sensor data are much more noisy than the simulated data in [60] due to the environmental variations across different days and different test sites. Data transformation is needed to better interpret the embedded information. Due to its denoising and time-frequency localization capability, wavelet transform is used in the personnel detection problem to preprocess the sensor data before constructing the cross machine.

Incorporation of wavelet transform into the framework of cross machine is not a straightforward task due to the issue of incompatible dimensions. A two-dimensional wavelet image is generated in the scale-shift domain by the wavelet transform of sensor time-series data. For the purpose of data-level information fusion, we need to extract the correlational information from wavelet images that are generated from different sensors. However, the number of scales used in wavelet transform may be different for each sensor, which results in incompatibility in the dimension of the wavelet images.

This section extends the concept of cross machine by incorporating wavelet transform into its framework and presents a method that resolves the issue of incompatible dimensions when fusing two wavelet images.

5.3.1 Conversion of Symbol Images to Symbol Sequences

The cross machines are constructed based on two symbol sequences \{s_1\} and \{s_2\} obtained from two different sensors (possibly heterogeneous) to capture the symbol level cross-dependence. The procedure of symbol image generation has been
introduced in Sec. 2.3. Conversion of the two-dimensional symbol image to one-dimensional symbol sequence while retaining the pertinent information is a difficult problem. One simple way for the conversion is to stack each row in the symbol image one after another. However, this method only works if the two symbol sequences \( \{s_1\} \) and \( \{s_2\} \) have the same number of scales in wavelet transform; otherwise their length of the generated symbol sequences will not be equal. More importantly, this method may suffer information loss in the frequency domain by only looking for relational dependency at the similar frequency bands in \( \{s_1\} \) and \( \{s_2\} \). It is highly possible that the low frequency component in \( H_1 \) has stronger correlation with the high frequency component in \( H_2 \).

To avoid these issues, a new method for converting symbol images from two sensors to symbol sequences for discovery of relational dependency is proposed. The key idea is to exhaustively find all possible combination of rows between the two symbol images [117]. A formal definition is as follows:

**Definition 5.3.1 (Conversion).** Let \( H_1 \) and \( H_2 \) be the wavelet coefficients (images) of sensor 1 and sensor 2, consisting of \( m_1 \times n \) and \( m_2 \times n \) pixels, respectively. Let \( W_j^i \subseteq H_i \) be the window that covers the \( j \)th scale in \( H_i \). Then the two symbol sequences \( \{s_1\} \) and \( \{s_2\} \) are defined as

\[
\{s_1\} = [S_\Sigma(W_{11}^1) \dots S_\Sigma(W_{11}^{m_1}) , \cdots , S_\Sigma(W_{12}^1) \dots S_\Sigma(W_{12}^{m_2})]
\]

\[
\{s_2\} = [S_\Sigma(W_{21}^1) \dots S_\Sigma(W_{22}^{m_2}) , \cdots , S_\Sigma(W_{21}^1) \dots S_\Sigma(W_{22}^{m_2})]
\]

where \( S_\Sigma(W_{ij}^i) \) is the symbol block in the \( j \)th row of symbol image \( i \). In other words, the first row of symbol image 1 is first repeatedly stacked for \( m_2 \) times in symbol sequence 1 while all the rows of symbol image 2 are stacked in symbol sequence 2. Both symbol sequences \( \{s_1\} \) and \( \{s_2\} \) have the length of \( n \times m_2 \) after this step. The process is repeated for all the remaining rows of symbol image 1.

By implementing the conversion procedure as defined above, the wavelet images \( H_1 \) of \( (m_1 \times n) \) pixels and \( H_2 \) of \( (m_2 \times n) \) are converted to one-dimensional symbol sequences of the same length \( (m_1 \times m_2 \times n) \). An illustration is given in Fig. 5.2, where \( m_1 = 1, m_2 = 2, n = 5 \) and \( |\Sigma_1| = |\Sigma_2| = 4 \).
5.3.2 Construction of Cross Machine

Cross dependencies are useful for data association among different sensors; sometimes the relations among targets are more important than the behavior of each target. Generation of joint statistics is difficult due to the curse of dimensionality. In traditional Bayesian theory, instead, the conditional statistics are modeled. Traditional time series analysis can handle linear relations efficiently (e.g., ARMA model), but for non-linear relations, it is computationally expensive. The concept of cross machine is introduced in [59, 60] to model the relational dependency among the sensor data from different modalities.

A formal definition of the cross machine is as follows:

**Definition 5.3.2.** A cross machine is a 6-tuple $R_{1\rightarrow 2} \triangleq (Q, \Sigma_i, \Sigma_o, \delta, q_0, \Psi)$, where

- $Q$ is a (nonempty) finite set, called set of states;
- $\Sigma_i$ is a (nonempty) finite set, called input alphabet;
- $\Sigma_o$ is a (nonempty) finite set, called output alphabet;
- $\delta : Q \times \Sigma_i \rightarrow Q$ is the state transition function;
- $q_0 \in Q$ is the start state;
- $\psi : Q \times \Sigma_o \rightarrow [0, 1]$ is an output mapping which is known as the output morph function and satisfies the condition $\sum_{\sigma \in \Sigma_o} \psi(q_j, \sigma) = 1$ for all $q_j \in Q$. The output morph function $\psi$ has a matrix representation $\Psi$, called the output (probability) morph matrix $\Psi_{ij} = \psi(q_i, \sigma_j), \forall q_i \in Q, \sigma_j \in \Sigma_o$. Note that $\Psi$ is a $(|Q| \times |\Sigma|)$ stochastic matrix.

Figure 5.3 depicts the concept of a cross machine that retrieves information via semantic cross compression of multi-modal sensor data as shown in Fig. 5.2. The
Figure 5.3. Cross machine constructed with symbol sequences in Fig. 5.2

76

Figure 5.3. Cross machine constructed with symbol sequences in Fig. 5.2

Goal is to predict the immediate future distribution of data sequence in sensor 2 (S_2), given the observed sequences in sensor 1 (S_1). In Fig. 5.3, the set of state are Q = {q_0, q_1, q_2, q_3}, the input alphabet is Σ_i = {a, b, c, d}, and the output alphabet is Σ_o = {0, 1, 2, 3}. The number on the digraph that connects the state and the output alphabet is the probability of observing the output alphabet at current state. For example, the probability of observing alphabet 0 in S_2 in the immediate future with the current state q_0 in S_1 is 0.5. The cross machine that predicts the distribution of S_2 given the observed sequence of S_1 is denoted as R_{1→2}, and the output morph matrix is called relational pattern. For the purpose of pattern classification, the stochastic matrix is transformed into an one dimensional feature vector with |Σ_i| × |Σ_o| elements.

5.4 Feature Fusion - Feature Selection

This section presents the fusion of features from cross machines that are constructed across different sensor pairs and the selection of a suitable set out of a pool of candidate features. A large number of candidate features are generated by the cross machine especially when the size of the input alphabet Σ_i and output alphabet Σ_o is large. Some of these features may give reliable class discriminative information while others do not carry any relevant information and hence, must be excluded as they could mislead the classifier. This task is not trivial since features that provide good classification information only achieve little improvement when combined in a feature vector, because of a high mutual correlation. Conversely, the correct combination of features with little class discriminative abilities may
Feature selection evaluates a subset of features as a group for suitability, and essentially divide into wrappers, filters, and embedded methods. A wrapper is a feature selector that convolves with a classifier, with the direct goal to minimize the classification error of the particular classifier. Usually, wrappers can yield high classification accuracy for a particular classifier at the cost of high computational complexity and less generalization of the selected features on other classifiers. Filter-based methods rank the features according to certain metric as a pre-processing step prior to the classifier, and select those features with high ranking scores. In practice, the filter approach has much lower complexity than wrappers; the features thus selected often yield comparable classification errors for different classifiers, because such features often form intrinsic clusters in the respective subspace. Embedded methods combine feature selection with the learning algorithm. The design of embedded method is tightly coupled with a specific learning algorithm, which in turn limits its application to other learning algorithms [128].

Evaluation of the feature subsets in the filter-based methods requires a scoring metric that grades a subset of features. Popular filter metrics include correlation, information theoretic measures (e.g., entropy, mutual information), and class separability (e.g., Euclidean/Mahalanobis distances, Fisher score).

Both wrapper and filter methods involve search algorithm. Search algorithm for filter-based methods is relatively simple since it only needs to rank the individual features using filter metric. As for the wrapper-based method, search algorithms are more complicated and can be broken into different categories, such as complete search, greedy sequential search, and randomized methods. Complete search method exhaustively looks for the best combination of features, and is generally impractical. Greedy sequential search iteratively evaluates a candidate subset of features, then modifies the subset and evaluates if the new subset is an improvement over the old. Depending on the search direction, greedy sequential search methods divide into greedy forward selection and greedy backward elimination. One of the most well-known randomized methods is the genetic algorithm (GA) that follows the principals of natural biological evolution.

In the personnel detection problem, datasets are usually collected from different sites over several days. Environmental changes may result in variations in the
extracted patterns. The wrapper method yields high classification accuracy at the cost of less generalization of the selected features. It is possible that the selected features perform well on the training pattern but generalize poorly to the testing pattern due to the fluctuation in the testing data and hence, filter-based method fits the personnel detection problem better due to its capability to generalize and its low complexity. In addition, in the cross machine technique, \( \binom{N}{2} \) relational patterns will be generated given \( N \) sensors. Some relational patterns may perform well if they capture the signature of the targets, whereas others may have lower performance if the relational patterns do not provide additional information. Another feature selection is needed to choose the sensor pairs with superior performance. Therefore, a cascaded filter structure, as shown in Fig. 5.4, is used in this chapter to select the sensor pairs and the features for the purpose of personnel detection. In Fig. 5.4, the first filter ranks the performance of all relational patterns generated by the cross machine, and the second filter selects the features from the output of the first filter. The implementation of the filter-based method is discussed in detail in Sec. 5.5.2.

5.5 Validation on Border Security Data

This section presents the experimental validation of the multi-level sensor fusion technique on border data. The data collection scenario and structure of the dataset are described, followed by the results of field data analysis.

5.5.1 Data Collection and Preprocessing

Experiments have been conducted in the U.S. southern border areas (Fig. 5.5(a)) by the U.S. Army Research Laboratory to collect data using multi-modal sensors,
Figure 5.5. Description of the experimental scenarios

including seismic, PIR and acoustic sensors (Fig. 5.5(b)). The data were collected in a realistic environment in an open field. There were three selected test fields in the area. These test fields were known to be used by the illegal aliens crossing the border. These places where the data were collected include: (a) wash (i.e., the dry bed of an intermittent creek), (b) trail (i.e., a path through the shrubs and bushes), and (c) choke point (i.e., a place where the targets are forced to go due to terrain difficulties). During multiple field tests, sensor data were collected for several scenarios that consisted of targets walking along an approximately 150 meters long trail, and returning along the same trail to the starting point, as shown in Fig. 5.5(c).

The targets consisted of people (e.g., male and female humans) and animals (e.g., horses). The humans walked alone and in groups with and without backpacks; the horses were ridden by humans and they made runs with walking or trotting. There were three sensor sites placed along the path with a spacing of 30 to 60 meters apart, each equipped with acoustic, PIR and seismic sensors. The seismic sensors were buried approximately 15 cm deep underneath the soil surface,
Table 5.1. Number of Feature Vectors for Each Target Class in the Data Sets

<table>
<thead>
<tr>
<th></th>
<th>Day 1</th>
<th>Day 2</th>
<th>Day 3</th>
<th>Day 4</th>
<th>Day 5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>6</td>
<td>29</td>
<td>19</td>
<td>38</td>
<td>28</td>
<td>120</td>
</tr>
<tr>
<td>Animal</td>
<td>12</td>
<td>12</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>72</td>
</tr>
</tbody>
</table>

the acoustic sensors were installed on a tripod above the seismic sensors, and the PIR sensors were collocated with the respective seismic and acoustic sensors. All targets passed by the sensor sites at a distance of approximately 5 m.

Multiple data runs were made to collect data sets of both classes. The data were collected over five days at different sites, and a brief summary is given in Table 5.1 that shows the number of runs of each class. Each dataset, sampled at a sampling frequency of $F_s = 2^{15}$ Hz, has 327680 data points that correspond to 10 seconds of the experimentation time. For computational efficiency, the original seismic, PIR and acoustic data were downsampled by a factor of 4.

Typical examples of the sensor signals are shown in Fig. 5.6, where the top row shows the multi-modal signals of a human walking scenario, and the bottom row shows those of an animal (a horse ridden by a man) walking scenario. It is difficult to distinguish the difference between Fig. 5.6(a) and Fig. 5.6(d) as well as Fig. 5.6(c) and Fig. 5.6(f), since the seismic signals and acoustic signals of human and horse walking are very similar. The amplitude of the signals is not a reliable feature to distinguish a human from a horse, because the amplitude of seismic and acoustic signals primarily depends on the distance between the target and the sensors rather than the type of the target. As for the PIR signals, the horse walking signal shown in Fig. 5.6(e) appears to be two humans walking out of phase. In Fig. 5.6(e), mode changes occur after the horse enters and leaves the field of view of the PIR sensor due to the change in radiation from above to below the atmosphere or vice versa. Although the difference between Fig. 5.6(b) and Fig. 5.6(e) is obvious, the scenario of a horse walking can be easily confused with the scenario of two humans walking out of phase. Therefore, identification of human and animal from sensor time-series data is a challenging task.
Figure 5.6. Typical examples of sensor signals collected from the field test

5.5.2 Results of Field Data Analysis

Figure 5.7 depicts the flow chart of the proposed multi-level fusion algorithm. Multi-modal sensors, including acoustic, PIR and seismic sensors, collect time-series data as the target passed by the sensor sites. The sensor data are preprocessed to remove the mean and normalize to unit variance. The normalization of all signals to unit variance makes the pattern classifier independent of the signal amplitude and any discrimination should be solely texture-dependent. Wavelet transform is performed on the data to discover the time-frequency domain information. The two-dimensional wavelet images are converted to one-dimensional symbol sequences following the procedure in Section 2.3. Cross machines are constructed on the symbol sequences from different sensor pairs. As shown in Sec. 5.4, feature selection is implemented in a cascaded filter structure. The first filter selects the best-performed relational patterns from all relational patterns generated by the cross machines. A \(k\)-nearest neighbors (\(k\)-NN) classifier [5] is used in the first filter as the filter metric, and the Fisher score [128][6] is used as the filter metric in the second filter. The selected feature are then classified by another \(k\)-NN classifier to determine its type.
Based on the spectral analysis of the ensemble of seismic and acoustic data at hand, a series of pseudo-frequency from the 1-20 Hz bands have been chosen to generate the scales for wavelet transform, because these bands contain a very large part of the footstep energy [116]. Similarly, a series of pseudo-frequency from the 0.2-2 Hz bands have been chosen for PIR signals to generate the scales. Upon generation of the scales, continuous wavelet transforms (CWT) are performed with appropriate wavelet basis function on the seismic and PIR signals. $db7$ is used for seismic signals since it matches the shape of seismic and acoustic signals very well, $db1$ is used for the PIR case since PIR signals are close to square waves. A maximum-entropy wavelet surface partitioning is then performed. Selection of the alphabet size $|\Sigma|$ depends on the characteristics of the signal: a small alphabet size is robust against noise and environmental variation, while a large alphabet size has more discriminative power for identifying different objects. The following parameters are used in this chapter: alphabet size $|\Sigma_{\text{seismic}}| = |\Sigma_{\text{pir}}| = |\Sigma_{\text{acoustic}}| = 8$, number of scales $|\alpha_{\text{seismic}}| = |\alpha_{\text{pir}}| = |\alpha_{\text{acoustic}}| = 4$.

The next step is to perform pattern classification on the feature vectors. All classifiers are implemented by $k$-NN classifier due to its simplicity. The available feature vectors are divided into two sets: a training set and a testing set. A five-way cross-validation [5] is used to generate the numerical results shown in this section. The data is divided into five sets by date and five sets of experiments are performed:

- Training: Day 1 + Day 2 + Day 3 + Day 4; Testing: Day 5
- Training: Day 1 + Day 2 + Day 3 + Day 5; Testing: Day 4
- Training: Day 1 + Day 2 + Day 4 + Day 5; Testing: Day 3
- Training: Day 1 + Day 3 + Day 4 + Day 5; Testing: Day 2

Figure 5.7. Multi-level fusion of multi-modal sensors for personnel detection
• Training: Day 2 + Day 3 + Day 4 + Day 5; Testing: Day 1

Training and testing on feature vectors from different days is very meaningful in practice. In each run of cross-validation, no prior information is assumed for the testing site or testing data. The classifiers’ capability to generalize to an independent data set is thoroughly tested in the five-way cross-validation.

The performance of individual sensors are examined by using the cross machine and benchmark methods. The cross machine that is constructed using the same sensor reduces to a simple PFSA, and the state transition matrices II are used as the feature vectors. For the purpose of comparative evaluation, kurtosis analysis [114], a benchmark technique of footstep detection, is used for target classification. Kurtosis analysis is useful for footstep detection because the kurtosis value is much higher in the presence of impulsive events (i.e., target present) than the case of no target [114]. To apply the kurtosis method to target classification, the kurtosis values of each training sample is computed and the optimal threshold in distinguishing human from animal is determined by searching. This threshold is then used for target classification in the testing stage. Cepstrum [129], which has been proven to be effective in speech recognition and acoustic signal classification, is used to extract the features from acoustic sensor data. The cepstral features are generated from each sample, and the first 500 coefficients are used for target classification. PIR sensor is widely used in motion detection; however, no benchmark technique is found in open literature that uses PIR for target classification.

Table 5.2 compares the results of using different methods for target classification. For seismic sensor, the SDF method (81.3%) outperforms the kurtosis method (58.9%). The cepstral features (70.8%) yield better result than SDF (58.3%) in analyzing the acoustic signals. This is because the data were collected over five different days and the acoustic sensors are vulnerable to environmental changes. In SDF, the scales generated based on the four-day training data may not be suitable for the testing data due to the environmental changes. Target classification using PIR sensor signals yields 72.9% accuracy.

Sensor fusion methods, including data fusion, decision fusion, and the proposed multi-level fusion method, are also compared with the methods that use single sensors in Table 5.2. Data fusion is implemented by constructing the cross machines and generating the relational patterns. The result of the pair of seismic and PIR
sensors is shown in Table 5.2 since this sensor pair yields the best classification accuracy among all sensor pairs in the training stage. Decision fusion is implemented by combining the labels generated by all the combinations of sensor pairs in the cross machine and making the decision by majority voting. The two-stage filter is used in the multi-level fusion to select the top features from the relational patterns that are generated in the data fusion step. Table 5.2 shows that the multi-level fusion method yields superior performance (85.4%) over other fusion methods and the methods using single sensors.

The confusion matrices of the classification results are shown below, where the rows are the ground truth of human and animal (in order) and the columns are the testing results of human and animal (in order).

\[
\begin{align*}
C_{\text{Kurtosis Seismic}} &= \begin{pmatrix} 92 & 28 \\ 54 & 18 \end{pmatrix}, & C_{\text{Cepstrum Acoustic}} &= \begin{pmatrix} 85 & 35 \\ 21 & 51 \end{pmatrix}, \\
C_{\text{SDF Seismic}} &= \begin{pmatrix} 104 & 16 \\ 20 & 52 \end{pmatrix}, & C_{\text{SDF Acoustic}} &= \begin{pmatrix} 109 & 11 \\ 41 & 31 \end{pmatrix}, & C_{\text{SDF PIR}} &= \begin{pmatrix} 81 & 39 \\ 41 & 31 \end{pmatrix}, \\
C_{\text{Data Fusion}} &= \begin{pmatrix} 107 & 13 \\ 20 & 52 \end{pmatrix}, & C_{\text{Decision Fusion}} &= \begin{pmatrix} 116 & 4 \\ 29 & 43 \end{pmatrix}, & C_{\text{Multi-Level Fusion}} &= \begin{pmatrix} 107 & 13 \\ 15 & 57 \end{pmatrix},
\end{align*}
\]

### 5.6 Conclusions and Future Work

This chapter presents and validates a multi-level fusion method for personnel detection, where cross machines have been used to extract the features from time series of multi-modal sensors. An appropriate selection of the basis function and the scale range allows the wavelet-transformed signal to be de-noised relative to the original noise-contaminated signal before partitioning of the wavelet image for symbol generation. The cross machine identifies the relational dependencies among
different sensors and mitigates the loss of significant information. A cascaded filter structure has been adopted to select the discriminative features for pattern classification.

The proposed method has been validated on a set of field data collected from different locations on different days. A comparative evaluation is performed on the feature vectors extracted from individual sensors, data fusion, decision fusion and multi-level fusion. Results show that, while seismic sensors alone and data-level fusion involving seismic sensors perform better than other sensors alone and other data-level fusion methods, the multi-level fusion method further improves the target classification performance.

While there are many research issues that need to be resolved before exploring commercial applications of the proposed method, the following topics are under active research:

- Development of algorithms to extract relational dependencies among three or more symbol sequences;
- Comparative evaluation of the proposed sensor fusion method with Dempster-Shafer and Bayesian network approaches [5] at the decision fusion level.
Chapter 6

Multi-Resolution Navigation of Autonomous Systems

Part II of this dissertation presents a novel multi-resolution navigation algorithm for autonomous system operating as a single agent in uncertain and complex environments. The algorithm seamlessly integrates the concepts of local navigation and global navigation, and enables adaptive decision-making based on the available spatial-temporal information. The algorithm enables adaptive decision-making and online replanning of paths. The algorithm also provides a complete coverage of the search space and does not suffer from the problem of having local minima, which is commonly encountered in potential-field-based methods. Chapter 6 presents the theoretical aspects of the proposed algorithm. The efficacy of the algorithm is validated on a high-fidelity Player/Stage simulator with various scenarios, as shown later in Chapters 7 and 8. Potential applications include terrain map generation, humanitarian demining, hazard detection and cleaning, and floor cleaning.

6.1 Introduction

Typical operations of autonomous systems include floor cleaning, lawn mowing, terrain-map generation, searching and rescuing targets from hazardous situations, cleaning of oil spills, and humanitarian de-mining. The capability of adaptive decision-making based on real-time information perception is a natural characteristic of biological systems. This capability is extremely crucial for enabling
distributed autonomy in uncertain environments where \textit{a priori} information is either incorrect or incomplete. Therefore, time-critical operations of autonomous systems require real-time decision-making that facilitates continuous adaptation to new information that is acquired \textit{in situ} by onboard sensing and pattern analysis devices. The new information refers to the observed phenomenon that relates to successful accomplishment of a mission objective (e.g., detection of a target), dynamic unfolding of the search area (e.g., detection of unknown obstacles and boundaries), and environmental changes (e.g., spreading of a hazardous chemical or wildfire). In particular, since autonomous systems are often required to operate in unknown environments without the benefit of \textit{a priori} known terrain-map, they must be capable of adapting their navigation trajectories quickly as the knowledge of the environment changes.

Technical literature abounds with diverse path planning algorithms that address the above problem to various degrees of autonomy [42]. Several algorithms have been proposed based on the \textit{artificial potential functions} [130][131], where the local potentials generate a \textit{virtual force field} (VFF) to navigate the autonomous systems. Potential field-based methods have been widely studied due to their simplicity and elegance; however, these methods have inherent problems such as trap situations (e.g., local minima and cyclic behavior) and no passage between closely spaced obstacles [132]. A variety of methods have been proposed to circumvent these problems [133][134][135][136]. Recent path-planning algorithms rely on \textit{sensor-based exploration} [137][138] that focus on real-time planning to reach a goal from a start point while avoiding obstacles based on sensor readings [139][140][141][142].

A large category of path-planning applications of autonomous systems require scanning of all points in the search area (e.g., mine sweeping) while dynamically discovering and avoiding obstacles at \textit{a priori} unknown locations. This is known as the \textit{Complete Coverage Problem} [51][143] and it is different from the problem of finding the shortest/optimum path to a specified goal. A variety of path-planning algorithms exist in technical literature for coverage of the search area using mobile sensors; a review of such algorithms has been presented by Choset in [144]. Recent coverage algorithms are based on Morse decomposition (cellular decompositions of the search space using critical points of the Morse functions) [53][54][55] and slice
Autonomous systems usually operate in dynamic environment, such as an exploratory vehicle is often required to move to a goal location without the benefit of a floorplan or terrain map. In this case, the autonomous system must be able to replan quickly as the knowledge of the environment changes. This is known as dynamic plan adaptation (DPA) (or simply dynamic adaptation). Many DPA algorithms exist [131, 145, 142], and many of them focus on avoidance of dynamic obstacles. However, in addition to dynamic obstacles, the autonomous system should be able to adapt to the target’s neighborhood as well. For example, floor cleaning robots should spend additional time to clean the neighboring area upon detection of the dirt since the probability of finding other dirt in this area is high. However, this issue has not been addressed in open literature.

The concepts of Statistical Mechanics [146], which were originally developed to study the ensemble properties of physical systems, have been extensively used for a diverse range of applications [147, 21, 148]. Specifically, the simple and powerful structure of Ising model [146] has an immense potential to model neighborhood dependencies between interacting elements of a complex system. These interactions, in turn, produce the collective global behavior through mutual interdependence. Technical literature abounds with diverse applications of Ising model [147, 21, 148]; however, the domain of applications of such models has so far been limited for extension to many other disciplines including the science of autonomy and artificial intelligence. A critical issue in autonomous systems (e.g., unmanned undersea vehicles (UUVs)) is to enhance onboard autonomy that facilitates in situ adaptation to contextual changes and that refers to the observed phenomenon of the environment (e.g., an event detection). Along this line, a navigation algorithm for dynamic adaptation in uncertain environments has been developed by Gupta et al. [149]; however, the issue of obstacle avoidance in uncertain dynamic environments was not addressed.

This chapter introduces a novel multi-resolution method for dynamic adaptation and complete coverage of a priori unknown environments using a single agent. A key feature of this method is that the underlying algorithm relies on the notions of local and global navigation that depend on the amount of spatio-temporal information needed for making the navigation decision. This feature supplies two
potential advantages to the algorithm: (i) the local navigation provides a reduced computational complexity in making real-time locally optimal navigation decisions via avoiding unnecessary global calculations and (ii) the global navigation, that are organized in a hierarchical manner, prevent the algorithm from being stuck into a local minimum. The algorithm switches from the local to the global navigation and vice versa as needed. These notions are inspired from biological systems that make instantaneous decisions based on the local sensory information and the long-distance decisions based on (partially) available global information.

6.2 Scope of the Research

The purpose of the research presented in Part II is to develop robust navigation algorithms for autonomous systems working in uncertain environments. The environment is partially known or unknown because it is costly and inflexible to require a complete map of the environment in which the autonomous system operates.

Map building is a key issue in this dissertation because of the requirement of unknown environments. The autonomous system should simultaneously construct a map with sensor information while covering the environment at the same time, which means the autonomous system has to make decisions based on a partial knowledge of the environment. Using a separate exploration phase for map building purposes is inefficient or even impractical in most cases.

An integral part of developing a path planning algorithm is to measure how well the algorithm performs in simulations or experiments. Performance metrics allow quantitative evaluation of an algorithm and thus enable comparisons between different algorithms. Despite the importance of quantitative metrics, this is an area that has received little attention. Development of suitable performance metrics is therefore another aim of this research.

This dissertation focuses on low-level decision-making and control of autonomous system in uncertain and complex environments. A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents within an environment, and can be used to solve problems that are difficult or impossible for a single agent to solve. However, most existing research on MAS addresses the issues of high-level decision and control in relatively simple environments. A MAS is not able to work
effectively and efficiently in real-world applications if a single agent cannot fulfill its tasks in complex environments. Therefore, this dissertation focuses on navigation of a single-agent autonomous system in uncertain and complex environments. The proposed algorithm in this dissertation can be extended to the case of MAS, which is a future research topic as discussed in Sec. 9.2.

6.3 Symbolic Grid Map: Enabling Adaptive Decision Making

Several robotic maps are commonly used in autonomous navigation tasks, including Voronoi diagram [150], cell decomposition [144, 54, 137, 55, 56], and grid map [151, 152]. These maps all employ some form of space decomposition, where a complex space is repeatedly divided until simple subregions of a particular type are created. Grid maps are a special case of space decomposition where the environment, both free space and obstacles, is decomposed into uniform grid cells. Emphasis has been placed on space decomposition based maps as they are the most common data structure used in coverage path planning. This is because coverage algorithms usually use the strategy of “divide and conquer”. Basically the environment is segmented into simpler subregions, and each subregion is then covered in turn [56]. Cell decomposition is favored by Choset in the area of robot coverage algorithms [144, 54, 137]; while grid maps are the most common choice among other researchers in the research community [151, 152].

The approaches described in this dissertation use grid maps to model the environment. Grid maps discretize the environment into so-called grid cells. Each cell stores information about the area it covers. Most frequently used are occupancy grid maps that store for each cell a single (either deterministic or probabilistic) value, representing the occupancy information of the grid cell. The advantage of grids is that they do not rely on predefined features which need to be extracted from sensor data. Furthermore, they offer a constant time access to grid cells and provide the ability to model unknown (unobserved) areas, which is an important feature in the context of exploration. However, they have the disadvantages of discretization errors and of requiring a lot of memory resources [152].
In most previous works, each grid cell in a grid map is assigned a value to represent the occupancy information, which is either deterministic (i.e., binary) or probabilistic [153, 152]. The occupancy information is used to distinguish the grid cells that are occupied by obstacle from the unoccupied grid cells. The path planning algorithm is then implemented in those unoccupied grid cells. For example, Zelinsky et al. proposed an offline coverage algorithm [154] based on the distance transform of a known grid map. In their work, the unoccupied grid cells are marked with the value obtained from the distance transform that represents a wavefront that propagates from the goal cell to the initial cell. Once the distance transform for the environment is calculated, the coverage path can then be formed by selecting the unvisited neighboring cell with the highest distance transform, starting from the initial cell. In essence, each grid cell does not only carry the occupancy information, but also contains navigation information for complete coverage. Nevertheless, the method introduced in [154] is for offline planning in known environment and is not efficient for online planning in unknown environment.

In the current work, the concept of symbolic grid map is introduced to represent the environment information in a single map. In symbolic grid map, each grid cell is assigned a symbol from an alphabet that represents all possible states the autonomous system may encounter during exploration. The symbols represent the states of i) unexplored, ii) explored, no obstacle and no target, iii) explored and obstacle present, iv) explored and target present. The details about these symbols are provided in Sec. 6.3.1. In floor cleaning application, the grid cells where dust is discovered can be marked as “target present” and cleaning robot will spend more time in cleaning these cells. In humanitarian demining, the grid cell where mine is detected is marked “target present”, and the vehicle should search the neighboring area of this cell for more mines due to the fact that mines are usually placed in a continuous pattern to block the access to a certain region.

This section presents the description and the formulation of the symbolic grid map. The search space is partitioned and a lattice system is constructed to form the symbolic grid map. Formulation of the search space is then extended to the multi-resolution sense to enable adaptive decision-making, which makes optimal navigation plans for autonomous systems depending on the spatio-temporal information available.
6.3.1 Search Space: Partitioning and Lattice Formulation

The environment to be explored is considered to be a planar area populated with a finite but unknown number of obstacles. The obstacles may have arbitrary shapes and sizes and their exact locations are a priori unknown. The terrain limits are defined either by a hard boundary (e.g., a wall) or by a soft boundary (e.g., subarea of a larger field). To this end, pertinent definitions are presented below.

Definition 6.3.1. (Partition) Let $S \subset \mathbb{R}^2$ be the search space. A partition $\mathcal{P}$ of $S$ is defined as $\mathcal{P} \triangleq \{P_\xi : \xi = 1, \ldots, |\mathcal{P}|\}$ such that $S$ is divided into mutually exclusive and exhaustive cells, i.e., $P_\xi \cap P_\nu = \emptyset \forall \xi \neq \nu$ and $\bigcup_{\xi=1}^{P} P_\xi = S$, respectively.

Definition 6.3.2. (State of a Lattice Site) Let $\Omega: \mathcal{P} \times \mathcal{T} \rightarrow \Sigma$ be a mapping that assigns each symbol in the alphabet $\Sigma$ to a cell of the partition $\mathcal{P}$ at each time epoch $t \in \mathcal{T}$, where $\mathcal{T}$ is the time span. Then, the state of a lattice site $\xi$ at time $t \in \mathcal{T}$ is defined as:

$$\gamma_\xi(t) \triangleq \Omega(P_\xi, t) \quad (6.1)$$

Definition 6.3.3. (Lattice System) The lattice system is defined as a set $\mathcal{L} \triangleq \{\mathcal{P}, \Gamma\}$, where $\mathcal{P}$ is a partition set of the search space $S$ and $\Gamma$ is the collection of the state of the lattice.

Remark 6.3.1. It follows from Definitions 6.3.1 and 6.3.2 that the partition $\mathcal{P}$ forms a grid of the search space $S$. The partition is constructed such that the dimensions of each element (i.e., cell) of the grid structure fall within the task area where the autonomous system carries out its oil spill cleaning task.

The search region is partitioned into a grid to form a finite-dimensional lattice structure such that each grid element (i.e., a cell) represents a lattice site. A generalized Ising model [149] is constructed over the lattice, which involves an exogenous time-varying potential function term to control the movement of the autonomous system in the search space. The construction of an energy potential of this spin model is similar to the pheromone of biological systems, which acts as a navigator to search for critical targets.
Once the spatial partitioning is done to construct a grid, the next step is to define a lattice, where each site of the lattice is isomorphic to a grid cell and represents a physical state of that cell. Therefore, the terms “grid cell” and “lattice site” are used interchangeably in this dissertation. Let $\Sigma \triangleq \{\sigma_j : j = 1, \ldots, |\Sigma|\}$ be a finite set of symbols, called the alphabet, which also determines all possible states for each partition cell $P_\xi \in \mathcal{P}$. For a standard Ising model [146], such an alphabet denotes the up and down states of the spin orientations. In this dissertation, the physical description of the state of each partition cell is described by an alphabet $\Sigma$ that is constructed with four possible symbols (i.e., $|\Sigma| = 4$). These symbols are defined as: $\sigma_1 = T$, $\sigma_2 = E$, $\sigma_3 = U$ and $\sigma_4 = O$, representing the following possible states of each partition cell: i) explored and target present, ii) explored and target not present, iii) unexplored, and iv) explored and obstacle detected, respectively. The term target refers to oil spill in Chapter 8. In essence, these states represent the four possible conditions of a partition cell. The objective here is to facilitate adaptive decision-making based on the observed states of the neighborhood.

The configuration space of the autonomous system at time $t \in \mathcal{T}$ is constructed as the Cartesian product:

$$\Gamma(t) = \bigotimes_{\xi=1}^{P} \gamma_\xi(t)$$

(6.2)

where $\Gamma$ is the collective state of the lattice that can have at most $|\Sigma|^{|\mathcal{P}|}$ possible state configurations, and $|\mathcal{P}|$ is the cardinality of the (finite) partition set $\mathcal{P}$.

As the autonomous system continuously searches different cells on the lattice, while moving from one cell to another, the collective state $\Gamma$ of the lattice unfolds in space and time by exhibiting different configurations that represent an evolution of the checkerboard pattern in the search space.

### 6.3.2 Multi-resolution Formulation of the Search Space

The local navigation area of an autonomous system is usually the local neighborhood of the current grid cell for computational efficiency. However, when there are no unexplored grid cells left in the local navigation area, the autonomous system has to increase the size of the neighborhood to find the possible unexplored grid cell
Figure 6.1. Multi-resolution formulation of the search space

Further away [155]. This method is not efficient, especially when the unexplored grid cells are far away from the autonomous system, the local neighborhood size needs to be increased to as large as the entire search space to cover these cells. This dissertation introduces the concept of multi-resolution navigation that partitions the search space at various levels of resolutions and uses the corresponding grid map for navigation according to the available spatio-temporal information.

The search space is coarsely and uniquely partitioned so that information can be consistently stored by the autonomous systems. The entire search space is approximately divided in half in both dimensions, keeping the fine grid cells intact, to form four cells. Then, the search space is divided by four in each dimension, again keeping cells intact and forming 16 cells in total. This procedure continues till the cell at the lowest level has the the same size or slightly smaller than the local neighborhood of the autonomous system. This size requirement is that the entire cell at the lowest level falls within the local neighborhood of an autonomous system and can be completely scanned at the local level. The level of the finest grid cells is denoted as level 0, the lowest level where each cell has about the same size with the local neighborhood is denoted as level 1. This procedure continues till the coarsest level that covers the entire search space. Figure 6.1 depicts the multi-resolution formulation of the search space.

Figure 6.2 shows the switch between the local and global navigation. In Fig. 6.2,
the search space is partitioned at three levels of resolution. The lattice system described in Section 6.3.1 corresponds to the level 0 that is the level of the finest resolution. The grid cells continues to merge to form level 1 and level 2. The autonomous systems first operates in level 0 till no unexplored grid cell remains in its local neighborhood. Then global navigation with level 1 is implemented to find unexplored cells, and the autonomous systems moves toward the centroid of the cell that has the most unexplored fine grid cells. If no unexplored grid cells are found, then the autonomous system continues to switch to the coarser level till unexplored cells are found. If no unexplored cells are found at the coarsest level, then the complete coverage task has been accomplished. As shown in Fig. 6.2, among the various levels, only the coarsest level covers the entire search space and has the real “global” view of the search space. This formulation avoids unnecessary global calculations and reduces the computational complexity in real-time implementation. Integration of local to global navigation is described in Sec. 6.4.

6.3.3 Autonomous Systems

Unlike several previous studies, this work assumes that the autonomous system occupies a finite area instead of being a point mass. For example, the autonomous system could be considered to have a circular body of diameter $D \in \mathbb{R}$ or a rectangular body of width $w \in \mathbb{R}$. It implies that the opening space between any two obstacles through which it can pass is lower-bounded by these parameters. Furthermore, the autonomous system is assumed to have the knowledge of its exact location at all times which can be drawn from a Global Positioning System (GPS) or an Indoor Localization System. The autonomous system is equipped with
navigation sensors (e.g., ultrasonic, infrared or LIDAR sensor) to detect obstacles within an area of radius $R \in \mathbb{R}$ around the robot’s body, as illustrated in Fig. 6.3.

As shown in Fig. 6.3, three areas are defined in the current formulation. The largest one is the navigation scan area, which is covered by the navigation sensors. The medium one is the local navigation area, where the locally optimal navigation decisions are made. The smallest one is the task area within which the autonomous system carry out its main task (e.g., target detection, oil spill removal, floor cleaning, and lawn mowing).

### 6.4 Algorithm of Multi-Resolution Navigation

This section presents the algorithms of multi-resolution navigation. The multi-resolution framework seamlessly integrates the algorithms of local and global navigation. The concept of generalized Ising model was proposed in [155] from which the potential of each grid cell was calculated. However, navigation merely based on the potential is unreliable and inefficient, especially in large and complex environment. In the current work, the generalized Ising model is used for local navigation, and a probability vector is used for navigation when there is no ideal or unexplored points within the local neighborhood. The author previously proposed expanding the neighborhood size [155], which can be extremely computationally expensive and time-consuming, especially in a large environment. The backtracking algo-
Algorithm 1 Multi-Resolution Navigation Algorithm

Input: current position $c(t) \in P(\mu)$, sensor readings $\{s_i(t)\}$, grid map $M(t)$

1: time step $t \leftarrow 1$
2: calculate the maximum coarse level $\ell_{\text{max}}$
3: loop
4: coarse level $\ell \leftarrow 0$
5: get current position $c(t)$ and sensor readings $\{s_i(t)\}$
6: update grid map $M(t)$ using $c(t)$ and $\{s_i(t)\}$
7: calculate the local energy term $E_{\xi_i}^C$, \forall cell $\xi_i \in N_{\kappa_0}(\mu)$
8: find grid cell $\xi^* = \text{argmax}_{i} E_{\xi_i}^C$
9: if $\exists$ a grid cell $\xi^*$ s.t. $E_{\xi_i}^C > 0$ then
10: while grid cell coordinates $\mu(t) = \mu(t - 1)$ do
11: $M(t + 1) \leftarrow \text{Local Navigation} (c(t), M(t))$
12: end while
13: repeat loop
14: else if $\ell < \ell_{\text{max}}$ then
15: while $\exists$ a cell $i$ at coarse level $\ell$ s.t. $p_i^\ell > 0$ do
16: $\ell \leftarrow \ell + 1$
17: end while
18: while grid cell coordinates $\mu(t) = \mu(t - 1)$ do
19: $M(t + 1) \leftarrow \text{Global Navigation} (c(t), M(t), \ell)$
20: end while
21: repeat loop
22: else
23: if $\exists$ a cell $i$ at coarse level $\ell$ s.t. $p_i^\ell > 0$ then
24: while grid cell coordinates $\mu(t) = \mu(t - 1)$ do
25: $M(t) \leftarrow \text{Global Navigation} (c(t), M(t), \ell)$
26: end while
27: repeat loop
28: else
29: exit loop
30: end if
31: end if
32: end loop

Algorithm proposed in [155] is also computation-intensive. Generating the probability vector is an effective way to circumvent these difficulties. The pseudo code and flowchart of the multi-resolution navigation algorithm are show in Algorithm 1 and Fig. 6.4, respectively.

As shown in the input of Algorithm 1, the autonomous system is assumed to have access to its current coordinate $c(t) \in P(\mu)$ with the grid cell coordinate $\mu$ and navigation sensor readings $\{s_i(t)\}$ at all times. The autonomous system stores the environment information by building the grid map $M(t)$. As shown in Fig. 6.4, the autonomous systems first check if there is any unexplored grid cells.
in its local neighborhood. If so, it performs local navigation; otherwise, it enters the global navigation mode. In both local navigation and global navigation, the autonomous system computes the goal when it enters a new grid cell instead of computing it at every step. This reduces the computation load while maintaining the adaptability to the environment changes. The switch between local navigation and global navigation is unidirectional, i.e., the coarseness of the level can only increase during the operation and is reset to the level 0 when it enters a new grid cell (see Figs. 6.1 and 6.2). It is noted that the switch can be made bidirectional at the expense of complexity and loss of robustness in real-time implementation.

Several important features of the multi-resolution algorithm are outlined in Sec. 6.5. The features include complete coverage of the uncertain environment, cleanup of all oil spills, no local minima problem, and low computational complexity.
6.4.1 Generalized Ising Model for Local Navigation

A four-state generalized Ising model is now constructed over the lattice system $L$ by extending the earlier work of Gupta et al. [149]. A local (implicitly time-dependent) energy term $E^L_\xi$ at a lattice site $\xi$ is defined as:

$$E^L_\xi(t) = \sum_{<\xi,\nu>_{\kappa_1}} J_{\xi\nu} \Psi(\gamma_\xi(t), \gamma_\nu(t)) + \Phi(B_\xi(t), \gamma_\xi(t))$$  \hspace{1cm} (6.3)

where $<\xi,\nu>_{\kappa_1}$ implies summation over a $\kappa_1$-neighborhood of $\xi$, for some $\kappa_1 \in \mathbb{N}$. The $\kappa$-neighborhood of a lattice site $\xi$ is defined as

$$\mathcal{N}_\kappa(\xi) = \{\nu : \max(|\xi_x - \nu_x|, |\xi_y - \nu_y|) \leq \kappa\},$$  \hspace{1cm} (6.4)

where $\xi_x, \xi_y \in \mathbb{N}$ and $\nu_x, \nu_y \in \mathbb{N}$ denote the $x$ and $y$ coordinates of lattice sites $\xi$ and $\nu$, respectively. The computation of Eq. (6.3) is carried out in $\mathcal{N}_{\kappa_0}(\mu)$, i.e., the $\kappa_0$-neighborhood of $\mu$, where $\mu$ is the grid cell occupied the autonomous system, and $\kappa_0$ is the distance between $\mu$ to the boundary of local navigation area.

The first term in the right hand side of Eq. (6.3) defines the total interaction potential due to the sum of the effects of neighbors on the state at a lattice site $\xi$. This term is called adaptation term because the effects of the observed states in the neighborhood cause changes in the resultant energy potential at a lattice site $\xi$, which enables real-time adaptation in the navigation path trajectory. The coefficient $J_{\xi\nu}$, denotes the interaction strength between two distinct lattice sites $\xi$ and $\nu$. For $\eta = \max(|\xi_x - \nu_x|, |\xi_y - \nu_y|)$, i.e., the distance between two neighborhood sites, $J_{\xi\nu}$ is given as

$$J_{\xi\nu} = \begin{cases} \eta^{-\alpha}, & \forall \xi \neq \nu \text{ and } \eta \in \{1, \cdots, \kappa_1\} \\ 0, & \text{otherwise} \end{cases}$$  \hspace{1cm} (6.5)

where $\alpha \in (0, \infty)$ is a control parameter. The (implicitly time-dependent) interaction function $\Psi$ is defined as

$$\Psi(\gamma_\xi(t), \gamma_\nu(t)) = \begin{cases} \psi_T, & \text{for } \gamma_\xi(t) = U, \gamma_\nu(t) = T \\ \psi_E, & \text{for } \gamma_\xi(t) = U, \gamma_\nu(t) = E \\ 0, & \text{otherwise} \end{cases}$$  \hspace{1cm} (6.6)
which implies the following conditions.

1. Any lattice site $\xi$ whose state is either $\gamma_\xi(t) = T$ (i.e., an explored site where a target is present) or $\gamma_\xi(t) = E$ (i.e., an explored site where no target is present) or $\gamma_\xi(t) = O$ (i.e., an obstacle) is not influenced by the state of the neighborhood. Therefore, the interaction potentials of sites that are in the above states are zero.

2. All neighborhood sites with a state $\gamma_\nu(t) = U$ (i.e., unexplored site) exert no influence on any lattice site because they provide no information to the neighborhood.

3. $\psi_T$ defines the influence of a site $\nu$ in the neighborhood with $\gamma_\nu(t) = T$ (i.e., an explored site where a target is present) on a site $\xi$ with $\gamma_\xi(t) = U$ (i.e., an unexplored site).

4. $\psi_E$ defines the influence of a site $\nu$ in the neighborhood with $\gamma_\nu(t) = E$ (i.e., an explored site where no target is present) on an unexplored site $\xi$.

The interaction function $\Psi$ has the following physical interpretation. A target detection at a certain lattice site causes distortion in the space-time potential field in the local neighborhood of the target resulting in an increase in energy by $\psi_T$, scaled to the interaction strength, thereby creating a dome-like structure. Therefore, as the autonomous system scans the area, the collective state $\Gamma$ of the lattice (see Eq. 6.2) unfolds in the space-time coordinates, and a detected target’s neighborhood with localized increase in energy becomes a high priority area. In this neighborhood, the unexplored sites (i.e., having a state $\gamma_\xi(t) = U$) with a high interaction potential tend to settle down to low energy states (i.e., explored sites). Thus, the autonomous system follows the high potential sites and, by scanning, turns them to low-energy states by conversion to explored sites that have no neighborhood interactions. If another target is detected in the neighborhood, it generates its own high interaction potential, which leads to constructive interference with the potential of an earlier target, and so on. Therefore, following high potential sites leads to an adaptation in the nominal trajectory of the autonomous system such that the high-priority areas are scanned earlier than the high-priority areas, thereby improving localized search performance. The interaction function
Ψ shows that the explored sites, where no target is detected, also exert relatively small influences on the neighboring unexplored sites, where the energy increases by a factor $\psi_E$. Therefore, $\Gamma$ tends to unfold in the neighborhood of explored sites with a priority, thereby generating a more uniform and orderly search. The construction of energy functional for adaptation of an autonomous system is conceptually similar to that of the chemical pheromone used for tracking by biological systems such as an ant.

The second term in the right hand side of Eq. (6.3) defines the navigation control function $\Phi$ that depends on an exogenous time-varying potential field $B_\xi(t)$ and the state $\gamma_\xi(t)$ at a lattice site $\xi$. The (implicitly time-dependent) function $\Phi$ is defined as:

$$\Phi (B_\xi(t), \gamma_\xi(t)) = \begin{cases} 
\phi_T, & \text{for } \gamma_\xi(t) = T \\
\phi_E, & \text{for } \gamma_\xi(t) = E \\
\phi_O, & \text{for } \gamma_\xi(t) = O \\
B_\xi(t), & \text{for } \gamma_\xi(t) = U 
\end{cases} \quad (6.7)$$

where the constants $\phi_T \leq 0$, $\phi_E \leq 0$ and $\phi_O < 0$ correspond to low-energy states of the explored sites in the presence (i.e., $\gamma_\xi(t) = T$) and absence (i.e., $\gamma_\xi(t) = E$) of a target and the presence of obstacle (i.e., $\gamma_\xi(t) = O$), respectively. As described earlier, the explored sites have zero neighborhood interaction potential and therefore, they settle down to these low energy states. On the other hand, the exogenous potential field $B_\xi(t)$ defines the time-varying potential at unexplored sites (i.e., $\gamma_\xi(t) = U$) and is given as

$$B_\xi(t) = B_\xi^* - C_{\xi,\mu}(t) \quad (6.8)$$

where $B_\xi^*$ represents the constant potential field that is constructed to navigate the autonomous system with no *in situ* adaptation. The relative cost potential function $C_{\xi,\mu}(t)$ defines the total decrease in potential at a grid cell $\xi$ due to travel and turn costs that are incurred to reach the grid cell $\xi$ from a current position $c(t) \in \mathcal{P}_\mu$ with the grid cell coordinate $\mu$ at time $t \in \mathcal{T}$. The cost function is constructed as
Algorithm 2 Local Navigation Algorithm

Input: current position $c(t) \in \mathcal{P}_\mu$, grid map $\mathcal{M}$
Output: updated grid map $\mathcal{M}$

1: calculate the local energy term $E_{\xi}^L$, $\forall$ cell $\xi \in \mathcal{N}_{\kappa_0}(\mu)$
2: set the centroid of grid cell $\xi^* = \arg\max_{\xi} E_{\xi}^L$ as the navigation goal
3: if front distance to obstacle $\leq$ threshold $d$ then
4: use Bug2 algorithm for obstacle avoidance
5: else
6: move toward the navigation goal
7: get current coordinate $c(t)$ and sensor readings $\{s_i\}$
8: update grid map $\mathcal{M}$ using $c(t)$ and $\{s_i\}$
9: end if

$$C_{\xi,\mu}(t) \triangleq \chi_{tr}T_{tr} + \chi_{tu}T_{tu}$$ (6.9)

where $T_{tr}$ is the cost of traveling across a single grid cell; $T_{tu}$ is the cost of turning; and $\chi_{tr}$ and $\chi_{tu}$ define the total number of grid cells and the total number of turns, respectively, that are required to reach $\xi$ from $\mu(t)$ along the shortest path. Equation (6.8) implies that the potential of a lattice site depends on the position and orientation of the autonomous system. For example, a far away site on the lattice with respect to the current position of the autonomous system will have less potential as compared to a nearby site, because of high traveling and turning costs, unless the far away site has a neighborhood target.

Therefore, Eq. (6.3) describes the total energy potential at a lattice site $\xi$, which is the sum of: i) neighborhood interaction potential due to nearby target locations, and ii) a time-varying field that depends on an externally applied potential and the traveling and turning costs. The autonomous system computes the values of the energy potentials $E_{\xi}^L(t)$ for all $\xi \in \mathcal{N}_{\kappa_0}(\mu)$, and sets the centroid of the grid cell $\xi^*(t)$ that has the highest potential as the goal for local navigation:

$$\xi^*(t) = \arg\max_{\xi \in \mathcal{N}_{\kappa_0}(\mu)} E_{\xi}^L(t)$$ (6.10)

The local navigation algorithm is summarized in Algorithm 2 using pseudo code. The Bug2 algorithm [156] is used as a subroutine for obstacle avoidance. An illustration of the Bug2 algorithm is given in Fig. 6.5, and the pseudo code of the Bug2 algorithm is shown in Algorithm 3. Many other obstacle avoidance algorithms are available, such as Bug 1 and TangentBug. The Bug2 algorithm is
Algorithm 3 Bug2 Algorithm

Input: start grid cell coordinate \( \mu_0 \), goal grid cell coordinate \( \nu \), grid map \( M \)

Output: updated grid map \( M \)

1: \( L_0 \leftarrow \) centroid of grid cell \( \mu_0 \); and \( i \leftarrow 1 \)
2: \( \text{loop} \)
3: \( \text{repeat} \) move on a straight line from \( L_{i-1} \) to centroid of goal grid cell \( \nu \)
4: \( \text{until} \) goal grid cell \( \nu \) is reached or obstacle is encountered at \( H_i \)
5: \( \text{if} \) goal grid cell \( \nu \) is reached \( \text{then} \)
6: \( \text{exit with success} \)
7: \( \text{end if} \)
8: \( \text{repeat} \) follow boundary
9: \( \text{until} \) (a) goal grid cell \( \nu \) is reached; or (b) \( m \)-line is re-encountered at \( Q \) such that \( Q \neq H_i \), \( d(Q, \nu) < d(H_i, \text{goal}) \), and line \( (Q, \nu) \) does not cross the current obstacle at \( Q \); or (c) \( H_i \) is re-encountered
10: \( \text{if} \) goal grid cell \( \nu \) is reached \( \text{then} \)
11: \( \text{exit with success} \)
12: \( \text{else if} \) \( H_i \) is re-encountered \( \text{then} \)
13: \( \text{exit with failure} \)
14: \( \text{else} \)
15: \( L_i \leftarrow Q; \ i \leftarrow i + 1 \)
16: \( \text{end if} \)
17: \( \text{get current coordinate} \ c(t) \text{ and sensor readings} \ \{s_i\} \)
18: \( \text{update grid map} \ M \text{ using} \ c(t) \text{ and} \ \{s_i\} \)
19: \( \text{end loop} \)

chosen over its counterparts for its simplicity in implementation and execution.

6.4.2 Probability Vectors for Global Navigation

Global navigation is usually operated over large search space that involves a large number of the finest-level grid cells. Calculation of the energy for all the grid cells requires high computation power, especially when the search space is large. This
Algorithm 4 Global Navigation Algorithm

Input: current position \( c(t) \in \mathcal{P}_n \), grid map \( \mathcal{M} \), level \( \ell \)
Output: updated grid map \( \mathcal{M} \)

1: generate grid map \( \mathcal{M}' \) of level \( \ell \) from grid map \( \mathcal{M} \)
2: compute probability vectors \( p^\ell_i \) for all cells at level \( \ell \)
3: set the centroid of cell \( i^* = \arg\max_i p^\ell_i \) as the navigation goal
4: if front distance to obstacle \( \leq \) threshold \( d \) then
5: use Bug2 algorithm for obstacle avoidance
6: else
7: move toward the navigation goal
8: get current coordinate \( c(t) \) and sensor readings \( \{s_i\} \)
9: update grid map \( \mathcal{M} \) using \( c(t) \) and \( \{s_i\} \)
10: end if

may undermine the real-time implementation capability of the algorithm and its further application in the multi-agent cooperation. To resolve this problem, a lightweight probability vector is used to store the environment information of each cell at the coarse levels. This vector mirrors the spins associated in an area of the environment in a probabilistic manner.

Each coarse cell partition is assigned a probability vector that records the spins of the fine cells located in it

\[
p^\ell_i = [|\gamma_\xi(t) = \sigma_1|, |\gamma_\xi(t) = \sigma_2|, |\gamma_\xi(t) = \sigma_3|, |\gamma_\xi(t) = \sigma_4|] \tag{6.11}
\]

where \( |\gamma_\xi(t) = \sigma_j| \) with \( j = 1, 2, 3, 4 \) signifies the number of fine cells in that region with spin \( \sigma_j \), and \( p^\ell_i \) refers to the \( i \)th cell of the \( \ell \)th level. Evidently, the sum of the four elements of the vector is the number of cells in the region. Then, the probability of finding a cell with given spin \( w \) in region \( i \) in level \( l \) is given by:

\[
p^\ell_i(\gamma_\xi(t) = w) = \frac{|\gamma_\xi(t) = w|}{\sum_{j=1}^4 |\gamma_\xi(t) = \sigma_j|} \tag{6.12}
\]

This probability vector is very lightweight and extremely easy to store. In global navigation, the probability vectors of all the grid cells at the current coarse level are calculated, and the one with highest probability of unexplored cells is set as the goal. The details of global navigation algorithm is summarized in Algorithm 4, where Bug2 algorithm \cite{156} is also used as a subroutine for obstacle avoidance.
6.5 Important Properties of the Algorithm

This sections outlines important features of the multi-resolution algorithm, which includes complete coverage of the uncertain environment, no local minima problem, and low computational complexity.

6.5.1 Complete Coverage

The multi-resolution algorithm provides complete coverage of the uncertain search space, as explained below.

Let there be an unexplored grid cell $\xi$ that is far away from the autonomous system. Then the autonomous system may switch to global navigation because no unexplored grid cells are within its $\kappa_0$-neighborhood. The coarse-level $\ell$ increases till a coarse-level cell $\varsigma$ with $\xi \in \varsigma$ and $p^\ell_\varsigma > 0$ is found, and the autonomous system navigates toward the centroid of the cell $\varsigma$. As the autonomous system approaches the fine-level grid cell $\xi$, the coarse level $\ell$ gradually decreases because a lower-level global view is large enough to cover the unexplored cell $\xi$. Eventually the grid cell $\xi$ falls within the $\kappa_0$-neighborhood of the autonomous system, and the autonomous system finishes the complete coverage task by moving into the grid cell $\xi$.

6.5.2 Local Minima Problem

The multi-resolution algorithm does not suffer from the local minima problem, as explained below.

Local minima problem occurs when there is no unexplored grid cell left in the local navigation area. In this case, the autonomous system switches from local navigation to global navigation and increases the coarse level till it finds unexplored grid cells. The autonomous system then moves toward the centroid of the coarse-level cell and resolves the local minima problem.

6.5.3 Computational Complexity

The multi-resolution algorithm has very low computational complexity that enables the real-time implementation of the algorithm, as explained below.
The computational complexities of the local and global navigation are evaluated separately. In the local navigation, the autonomous system searches for the grid cell with highest energy within a square area of length $2\kappa_0 + 1$, which has $(2\kappa_0 + 1)^2 - 1$ grid cells excluding the grid cell occupied by the autonomous system. Therefore, the local navigation algorithm has a computational complexity of $O(\kappa_0^2)$. In the global navigation, as illustrated in Fig. 6.2, the autonomous system only needs to search a $2 \times 2$ area, no matter which coarse level it is at. Overall, the proposed algorithm has a computational complexity of $O(\kappa_0^2)$. Usually $\kappa_0 \simeq \frac{1}{10} L$ suffices full coverage of a local neighborhood, where $L$ is the length of the search area; so the proposed algorithm has a low computational complexity.

6.6 Validation Test-Bed: Player/Stage

Player/Stage project is one of the most widely used robotics software package, which consists of libraries that provide access to communication and interface functionality. The robot server Player provides an architecture where many modules can be independently written and connected through a custom middleware relying on transmission control protocol (TCP) communication. Stage is a lightweight, highly configurable robot simulator that supports large populations. All sensor and actuator models are available through Players standard interfaces [85].

A Pioneer 2AT robot is modeled in the simulator as the autonomous system. The Pioneer robot is equipped with sonar sensors for navigation. The range of
the sonar sensors is 5 meters, and in total 16 sonar sensors are installed around the Pioneer robot. The robot has access to its position information in real time. The robot is also equipped with oil detection sensor that is implicitly modeled. Kinematic constraints were modeled in the Pioneer robot, such as minimum turn radius, top speed and maximum acceleration. A screenshot of the Stage simulator is shown in Fig. 6.6, where the Pioneer robot is exploring an uncertain environment with obstacles. Although the Pioneer robot is used for validation, the navigation algorithm presented in the dissertation is meant to be generic and not platform dependent.
Most path planning algorithms target finding the shortest or quickest path from one point to another. However, in many applications, a coverage path is needed instead. The aim of a coverage path planner is to create a path that covers the entire search space. In other words, given an initial location, it does not matter where the final location is, as long as the journey visits the entire search space [56]. Examples of tasks that require a coverage path include cleaning [50], surface coating [157], and demining [49]. Figure 7.1(a) shows an example of home floor cleaning using iRobot Roomba, Fig. 7.1(b) shows an example of humanitarian demining using a mine flail in Sudan [158], and Fig. 7.1(c) shows an example of naval mine countermeasure using a surface vessel [159].

Coverage path planning is similar to exploration, but not exactly the same. When exploring, an autonomous system sweeps its long range sensors, moving so as to sense all of its environment, often to build a map. The autonomous system may spend more time in exploring certain areas that have higher priority or richer information to be gained. In the coverage application, the autonomous system is required to pass over all the points in the search space.

Autonomous systems may operate in known and unknown environments. For a given robotic map, there are many ways to conduct a search and plan a path. For example, there are numerous search algorithms for graphs, such as A* search and depth-first search [47]. In many applications, the autonomous system are not
Figure 7.1. Typical examples of tasks that require complete coverage of unknown environments: (a) floor cleaning robot, (b) landmine removal using a mine flail in Sudan, and (c) naval mine countermeasure.

Provided with a map, and they have construct a map for path planning using sensor information. There are two distinct ways to handle this situation [56]. The first method is to carry out exploration phase to construct an accurate map [160] before any path planning is done. The alternative is to make assumptions concerning the uncertain areas in the map in order to commence path planning, and then update the planned path whenever new environmental information becomes available [161]. In other words, path planning is done on a partial map of the environment. Using a separate exploration phase for map building purposes is considered inefficient because a coverage path already requires the autonomous system to visit the entire search space. This is impractical in many time-critical missions such as humanitarian demining and oil spill cleaning.

This chapter validates the multi-resolution algorithm proposed in Chapter 6 on various complex scenarios that are inspired from real life. The environments which the autonomous systems operate in is assumed to be static, i.e., the locations of obstacles and targets in the environments do not change with time. The issue of operation in dynamic environments is discussed separately in Chapter 8.

7.1 Multi-Resolution Navigation Algorithm for Complete Coverage

The multi-resolution navigation algorithm is described in Sec. 6.4 in detail. The local energy term $E_L^c$ comprises two parts, namely, the adaptation term and the
navigation term. The adaptation term defines the total interaction potential due to the sum of the effects of neighbors on the state at a lattice site $\xi$. The adaptation to the neighborhood will lead to deviation from the current searching pattern.

In the complete coverage problem, it is important to maintain the regularity of the trajectory such that the autonomous system is able to cover the entire search space with minimum amount of turns. It is desirable for the autonomous system to follow the offline navigation plan defined in zigzag pattern and make online replanning of the path only if obstacles are detected in the original path. Therefore, the adaptation term in Eq. (6.3) is disabled in the complete coverage problem, and Eq. (6.3) reduces to

$$E_\xi^L(t) = \Phi (B_\xi(t), \gamma_\xi(t))$$

(7.1)

where $\Phi$ and $B_\xi$ are defined similarly with those in Chapter 6. An additional constraint is also imposed to the local navigation algorithm, and the revised local navigation is defined as follows.

It is seen that lines 4-9 in Algorithm 5 and lines 1-6 in Algorithm 2 are identical. In Algorithm 5, additional statements are added to check if there are unexplored grid cells adjacent to the current grid cell in both upward and downward directions. If the unexplored grid cells are found, the autonomous system will then move toward the topmost or the bottommost unexplored grid cell. This is to make sure the autonomous system starts exploration of a column either from the top or from the bottom such that a trajectory that is close to zigzag pattern is generated. Without this constraint, the autonomous system may start the exploration in the middle of a column and enter a new column upon finishing the upper/lower part of the column. As a result, the system has to make a turn and returns to the original column to clean up the remaining grid cells when it passes by the starting point. This change in the trajectory will spread and affect the search pattern in the remaining area. The autonomous will follow the new search pattern and make unnecessary turns in each of the remaining columns. The additional constraint added to the local navigation avoids unnecessary turns of the autonomous system, thereby saving energy and maintaining the regularity of the search pattern.

The remaining part of the multi-resolution algorithm used in this chapter is
Algorithm 5 Local Navigation Algorithm for Complete Coverage

**Input:** current position $c(t) \in P_\mu$, grid map $M$

**Output:** updated grid map $M$

1: search upward for unexplored cells till a non-unexplored cell is reached, then record the number of unexplored cells $n_{up}$ and the coordinate of the last unexplored cell $\nu_{up}$

2: search downward for unexplored cells till a non-unexplored cell is reached, then record the number of unexplored cells $n_{down}$ and the coordinate of the last unexplored cell $\nu_{down}$

3: if $n_{up} = 0$ and $n_{down} = 0$ then

4: calculate the local energy term $E_{L_{\xi}}$, $\forall$ cell $\xi \in N_{\kappa_0}(\mu)$

5: set the centroid of grid cell $\xi^* = \arg\max_i E_{L_{\xi}}$ as the navigation goal

6: if front distance to obstacle $\leq$ threshold $d$ then

7: use Bug2 algorithm for obstacle avoidance

8: else

9: move toward the navigation goal

10: end if

11: else

12: if $n_{down} \geq n_{up}$ then

13: move toward the centroid of grid cell $\nu_{down}$

14: else

15: move toward the centroid of grid cell $\nu_{up}$

16: end if

17: keep the states of the unexplored grid cells unchanged

18: end if

19: get current coordinate $c(t)$ and sensor readings $\{s_i\}$

20: update grid map $M$ using $c(t)$ and $\{s_i\}$

the same with that defined in Chapter 6.

### 7.2 Performance Metrics of Complete Coverage

An integral part of developing a robotic coverage algorithm is to measure how well the algorithm performs in experiments. Most commonly in complete coverage work, results are presented qualitatively, showing pictures of the route taken by the robot from simulation [154, 162, 163], or from real robot experiments [54]. Performance metrics allow quantitative evaluation of implementations. They also permit comparisons between different algorithms.

Wong raises two questions regarding the coverage operations in [164]. Firstly, how much of the environment is covered or missed? Secondly, how much time is wasted on revisiting the area that has been covered already?

The first question can be answered with a measure of the effectiveness of the operation. In simulation, this is commonly measured by calculating the percent-
age of grid cells covered [165]. In real robot experiments, existing approaches to estimating percentage coverage include sprinkling sawdust [166] and using the coverage factor [167]. Neither methods produce a good estimate of the percentage coverage for real robot experiments [56].

The second question is an inquiry about the efficiency of the operation. For simulated experiments, Gabriely et al. uses the number of repeatedly covered grid cells [45]. There are two minor flaws with using repeatedly covered cells as a metric. Firstly, a repeatedly covered cell maybe covered more than twice. Secondly, the figure is not normalized against the total number of grid cells in the environment. coverage experiments.

Wong [56] proposes two metrics that overcome the flaws in the previous works [165, 166, 167]. These metrics are briefly reviewed in this section and are used to measure the performance of coverage experiments in this dissertation.

### 7.2.1 Effectiveness: Percentage Coverage

The effectiveness of a coverage algorithm is the amount of the total search space covered by a autonomous system running the algorithm. Therefore,

\[
C = \frac{\text{area of search space covered}}{\text{total reachable area of search space}} \quad (7.2)
\]

The coverage metric \(C\) calculates the percentage coverage of an experiment. It requires the estimation of the area of the search space covered in an experiment and a measure of the area of the reachable search space in the environment [56].

The simulation environment is assumed to be uniform grid with square cells. Therefore, area in simulation can be measured in numbers of grid cells. Equation (7.2) can be rewritten as

\[
C = \frac{\text{number of covered grid cells}}{\text{number of reachable grid cells}} \quad (7.3)
\]

where the number of reachable grid cells can be calculated using flood fill algorithm. The number of covered grid cells is found by counting the explored grid cells.
7.2.2 Efficiency: Path Length

Wong proposed the metric *path length* as an indication of the efficiency of a coverage algorithm [56]. Path length is defined as follows

\[ L = \frac{\|P_a\|}{\|P_m\| \times C} \tag{7.4} \]

where \(\|P_a\|\) is the actual path length traveled by the autonomous system, \(P_m\) is called the *minimal path* since it is the shortest coverage path for an autonomous system that can “teleport” with no cost associated with the teleport operation, and \(C\) is percentage coverage as defined in Eq. (7.3). The length of the actual path \(P_a\) can be obtained from the autonomous system’s GPS record or encoder readings. The minimal path \(P_m\) depends on the configuration of the environment.

For implementation in simulation, the path length measure from Eq. (7.4) is then reformulated as

\[ L = \frac{\text{number of moves}}{\text{number of reachable grid cells} \times C} \tag{7.5} \]

The actual path length is calculated as the number of moves the autonomous system has made. This is similar, but not the same as the number of grid cells in the autonomous system’s path. The difference is in the handling of the grid cells that are visited multiple times. In Eq. (7.5), a grid cell that has been visited \(n\) times occupies \(n\) steps in the path.

The number of reachable grid cells is used as the length of the minimal path. This comes directly from the definition of the minimal path, which is the path length needed to achieve complete coverage with a teleport robot. With no cost to teleport, such a robot can cover \(n\) grid cells in \(n\) steps [56].

7.3 Case Studies in Uncertain Environments

In this section, five complex scenarios are used to examine the performance of the multi-resolution algorithm. The environments in all four scenarios used in this chapter are static and *a priori* unknown to the autonomous system. The dimension of the maps is 50\(m\) \(\times\) 50\(m\). The size of the grid cell at the finest level is 1\(m\) \(\times\) 1\(m\).
which is slightly larger than the size of the autonomous system. The starting point of the search is the bottom left corner. The typical zigzag pattern is optimal for search an area in terms of minimum number of turns when no adaptation to obstacle is needed. Therefore, for area coverage planning, the exogenous potential field $B^*_\xi$ in Eq. (6.8) is designed for back and forth motion such that the potential field has a decreasing magnitude from column to column, starting from a maximum value of magnitude 25000 at the start point, while having equipotential sites on each column. The other parameters in Eqs. (6.3) to (6.9) have been selected to be $\kappa_0 = 3, \phi_E = 0, \phi_T = 0, \phi_U = 20000, T_{tr} = 100$, and $T_{tu} = 150$ in the simulation exercises; however, the specific values of these parameters do not have very significant effects on the algorithm performance as long as they are of the same relative order of magnitude.

7.3.1 Cave with Maze and Obstacles

This experiment is designed to test the performance of the proposed algorithm in exploring a maze and then escaping from it. Large obstacles, both isolated in the middle and attached to the boundary, are also used in this experiment to increase the complexity of the map. Solid walls are attached to the boundary of the search space in three directions to mimic a cave environment.

The sonar sensors only detect the contour of the obstacle and are not able to penetrate the obstacle. As a result, even though the area surrounding the obstacle is explored and the contour of the obstacle is closed, there still appears to be unexplored grid cells inside the obstacle. The autonomous system will try all means to reach these unexplored grid cells although they are not reachable. This issue is resolved by using the flood-fill algorithm that automatically fills up the cavity inside an object when its contour is closed.

The simulation result is shown in Fig. 7.2, where the first row shows the motion trajectory of the autonomous system and the second row shows the environment maps. In the first row of Fig. 7.2, the robot is represented by the blue dot, and the red circles indicates the motion trajectory. In the second row of Fig. 7.2, the color coding is defined as follows: light green for $\sigma_1 = U$ (unexplored), medium green for $\sigma_2 = E$ (explored and no obstacle), dark green for $\sigma_4 = O$ (obstacle
(a) start exploration  (b) wall following  (c) cavity filling  (d) complete coverage

Figure 7.2. Example of complete coverage in a map with a maze and a large obstacle detected), yellow for the buffer around the obstacle. The buffer was added to avoid the collision between the robot and obstacles since the robot has a minimum turn radius. The yellow cells are given the same cost as the obstacles, and can be considered as part of the obstacles.

Four snapshots are shown in Fig. 7.2(a)-(d). In Fig. 7.2(a), the autonomous system start exploring the unknown environment, following the potential field $B_\star^\xi$ assigned offline. While performing back and forth search, the robot entered the maze and then used Bug2 algorithm to exit the maze, as shown in Fig. 7.2(b). Fig. 7.2(c) shows the moment when the flood-fill algorithm was implemented once the contour of the large obstacle was closed. Fig. 7.2(d) shows the moment when the robot was about to reach the last unexplored grid cell. The cavity in the obstacle attached to the boundary was also filled up by the flood-fill algorithm.

7.3.2 Apartment

This experiment aims to validate the proposed algorithm in a real-life floor-cleaning scenario. A map with the layout of a typical apartment was designed. The apartment consists of four rooms with different size, an open region in the center as the living room, and obstacles of different shape and size as furniture.

Four snapshots are taken and shown in Fig. 7.3(a)-(d). Fig. 7.3(a) shows the
starting moment when the autonomous system re-enters the room at the left bottom corner to finish the remaining area. Fig. 7.3(b) shows the moment when the autonomous system explores the concave obstacle. Fig. 7.3(c) shows the autonomous system cleaning up the room in the right upper corner after finishing the living room, and Fig. 7.3(d) shows the robot cleaning up the last grid cell.

It is seen in Fig. 7.3 the proposed algorithm is able to solve the complete coverage problem with minimum overlapping trajectory. The overlapping trajectory is due to obstacle avoidance and switch between rooms. A potential application of the algorithm is autonomous floor-cleaning robot. A significant advantage over the commercial available products (e.g., iRobot roomba/scooba) is the proposed algorithm guarantees complete coverage of complex environment with minimum overlapping trajectory, which in turn saves operation time and energy, and is very meaningful for the cleaning task consuming washing fluid. The only assumption made in the algorithm is that the robot has access to the location information. This can be achieved by indoor localization system, such as Hagisonic StarGazer robot localization system.
7.3.3 Obstacle-Rich Environment

This experiment is designed to further test the performance of the multi-resolution navigation algorithm in obstacle-rich environment. As shown in Fig. 7.4, obstacles of various shapes and sizes are scattered in the search space. Similar with previous experiments, four snapshots are taken and shown in Fig. 7.4.

Fig. 7.4(a) shows the starting moment when the autonomous system follows the offline navigation plan to search the environment using a zigzag pattern. Fig. 7.4(b) shows the moment when the autonomous system finishes exploring the interior of a concave obstacle. Fig. 7.4(c) shows the autonomous system using Bug2 algorithm to escape from a U-shape obstacle and looking for unexplored grid cells. Fig. 7.4(d) shows the autonomous system cleaning up the last grid cell.

7.3.4 Large Obstacles

This experiment targets testing the capability of the proposed algorithm in handling the situation of large obstacles. As shown in Fig. 7.5, a large obstacle is located in the center of the map and a small obstacle is located in the upper left corner. As mentioned in Sec. 7.3.1, since the sonar sensors are not able to penetrate the obstacle, a contour of the large obstacle is generated with its interior
filled with unexplored cells. These unexplored cells must be marked as obstacle to prevent the autonomous system from moving toward the obstacle. Flood fill algorithm is used to fill up the cavity inside the obstacles when its contour is closed. To improve the computational efficiency, the flood fill algorithm is implemented every a hundred steps in the simulation.

The simulation result is shown in Fig. 7.5. Figure 7.5(a) shows the starting moment when the autonomous system follows the zigzag pattern. Part of the obstacles have been detected by the sonar sensors. In Fig. 7.5(b), the autonomous system moves toward the other side of the large obstacle through the narrow passage between the obstacle and the boundary of the search area. Figure 7.5(c) shows the autonomous system detects the major part of the large obstacle’s contour and cleans up the area around the small obstacle that has higher potential. In Fig. 7.5(d), both cavities inside the two obstacles have been filled up by the flood-fill algorithm, and the autonomous system is cleaning up the last unexplored grid cell in the upper right corner.
7.3.5 Spiral Maze

This experiment aims to show how the design of the offline navigation plan $B^*_\xi$ affects the efficiency of searching in the complete coverage problem. In the previous four experiments, a potential field has a decreasing magnitude from column to column is designed to make the autonomous system to follow the zigzag searching pattern. This commonly used searching pattern, however, may not be the most efficient searching pattern in some circumstances. In this experiment, the autonomous system is required to explore a spiral maze and cover the entire search space. Figure 7.6 shows two searching patterns designed for this experiment, namely, zigzag pattern and spiral pattern. It is intuitive that the spiral searching pattern, as shown in Fig. 7.6(a), matches the shape of a spiral maze much closer than the zigzag pattern. The efficiency of these two searching patterns are evaluated in a spiral maze in two separate runs. The simulation results are shown in Figs. 7.7 and 7.8, where the autonomous system follows the zigzag pattern in Fig. 7.7 while it follows the spiral pattern in Fig. 7.8. Four snapshots are taken and shown in both figures.

Figure 7.7(a) shows the autonomous system starts exploration following the zigzag pattern, and Fig. 7.7(b) shows the autonomous system explores the interior of the maze. As shown in Fig. 7.7(c), the autonomous system uses wall following algorithm to get out of the maze when all the grid cells inside the maze are explored. Figure 7.7(d) shows the moment before the complete coverage goal is achieved. It
is seen in the top part of Fig. 7.7(d) that the autonomous system makes frequent
turns to cover the grid cells outside the maze, due to the zigzag searching pattern.
In addition, the trajectory of the autonomous system inside the maze is very
dense because the trajectory made by the autonomous system during wall following
overlaps its previous trajectory. The unnecessary turns and overlapping trajectory
significantly increases the time and the energy to cover the spiral maze.

Figure 7.8 shows the results using the spiral searching pattern. Compared to
Fig. 7.7, the trajectory in Fig. 7.8 is much more regular and has fewer overlapping
part. Figure 7.8(a) shows the autonomous system follows the spiral searching
pattern and explores the exterior of the maze. Figure 7.8(b) shows the moment
when the autonomous system enters the maze and all the grid cells outside the maze
are explored. Figure 7.8(c) shows the autonomous system continues to explore
the interior of the maze, and Fig. 7.8(d) shows the autonomous system finishes
complete coverage of the search space.

As shown in Figs. 7.7 and 7.8, both the zigzag pattern and the spiral pattern
are able to achieve the goal of complete coverage, i.e., percentage coverage $C_{zigzag} =
C_{spiral} = 1.00$. These figures also qualitatively show that the spiral pattern is more
efficient than the zigzag pattern. Their efficiencies are evaluated quantitatively in
the following section.
7.4 Quantitative Evaluation

This section uses the performance metrics described in Sec. 7.2 to quantitatively evaluate the performance of the proposed algorithm in the experiments discussed in Sec. 7.3. The first four experiments in Sec. 7.3 are named case 1 through case 4 in order. In the spiral maze example shown in Sec. 7.3.5, the simulation using zigzag pattern is named case 5, and the one using spiral pattern is named case 6.

Two performance metrics are introduced in Sec. 7.2. The concept of \textit{percentage coverage} indicating the effectiveness of the algorithm is defined in Eqs. (7.2) and (7.3). The concept of \textit{path length}, defined in Eq. (7.5), indicates the efficiency of the algorithm in finishing the complete coverage task. In addition to these metrics, the time taken to finish the complete coverage task, $t$, is also used as a metric.

The results of the quantitative evaluation of the algorithm in different cases are shown in Table 7.1. It is seen in the table that the proposed algorithm achieves the goal of complete coverage in all of the six cases, i.e., $C = 1.00$. The time taken to finish the complete coverage $t$ and the path length $L$ are similar among these cases except case 5. The path lengths in cases 1-4 and case 6 are all smaller than 1.1, indicating the trajectory generated by the proposed algorithm is very close to the minimal path. The results of case 5 are anticipated because the searching
Figure 7.9. This history of percentage coverage is shown as a function of time for all six cases in Fig. 7.9. Figure 7.9(a) shows the results of cases 1-4, and Figure 7.9(b) shows the results of cases 5 and 6. The algorithm takes different amounts of time to cover the entire search space, depending on the complexity of the environment. It is seen in Fig. 7.9(b) that it takes 60295 steps for the autonomous system to finish the complete coverage if the zigzag pattern is used, and the number of steps is reduced by half if the spiral pattern is used. One step approximately corresponds to 0.20 sec of real time in simulation.

Table 7.1. Quantitative Evaluation of the Proposed Algorithm in Different Experiments

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
<th>Case 4</th>
<th>Case 5</th>
<th>Case 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>41504</td>
<td>44429</td>
<td>46051</td>
<td>33437</td>
<td>60295</td>
<td>36974</td>
</tr>
<tr>
<td>$C$</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>$L$</td>
<td>1.099</td>
<td>1.072</td>
<td>1.073</td>
<td>1.083</td>
<td>1.343</td>
<td>1.007</td>
</tr>
</tbody>
</table>

pattern used in this case does not suit the shape of the maze.

The history of percentage coverage is shown as a function of time for all six cases in Fig. 7.9. Figure 7.9(a) shows the results of cases 1-4, and Figure 7.9(b) shows the results of cases 5 and 6. The algorithm takes different amounts of time to cover the entire search space, depending on the complexity of the environment. It is seen in Fig. 7.9(b) that it takes 60295 steps for the autonomous system to finish the complete coverage if the zigzag pattern is used, and the number of steps is reduced by half if the spiral pattern is used. One step approximately corresponds to 0.20 sec of real time in simulation.

7.5 Summary and Discussion

This chapter presents the application of the multi-resolution navigation in the complete coverage problem, where the autonomous system is required to cover the entire search space. An additional constraint is imposed to the local navigation algorithm to maintain the regular trajectory and thus reduces unnecessary turns in operation. Five complex scenarios are used in this chapter to examine
the performance of the proposed algorithm. The validated scenarios include cave, apartment, obstacle-rich environment, large obstacles, and spiral maze. Performance metrics, including percentage coverage ($C$), path length ($L$), and required time to finish the task ($t$), are used to quantitatively evaluate the performance of the proposed algorithm. Simulation results indicate the proposed algorithm is very effective and able to accomplish the goal of complete coverage (i.e., $C = 1.00$) in all the cases. The results also show that by using proper searching pattern, the proposed algorithm is very efficient and the path length is very close to that of the minimal path.
In the previous chapter, it is shown that the multi-resolution algorithm is effective and efficient in implementing navigation tasks for autonomous systems, such as complete coverage of unknown and static environments. However, the environment is dynamic in many cases, such as the spreading and drift of the oil spills. The algorithm presented in the previous chapter is unable to adapt to the changes and is thus inadequate for accomplishing the task. This chapter extends the work presented in the previous chapter and introduces the capability of dynamic adaptation that, upon detection of the targets, would enable the autonomous system to replan its actions online. The work in this chapter is presented in the context of oil spill cleaning, which is a typical application of autonomous systems.

8.1 Motivation

The recent Deepwater Horizon oil spill in the Gulf of Mexico has attracted the attention of world community due to its colossal ecological, economic and social impacts. Over 210 million gallons of crude oil was released and the slicks and sheen of the surface oil directly affected over 180,000 square kilometers of ocean surface [168]. Figure 8.1 shows the lingering oil slick illuminated by sunlight off the Mississippi Delta on May 24, 2010. In order to clean this oil spill, over 39,000 personnel, 5,000 vessels, and 110 aircraft were involved, over 700 km of booms
have been deployed, 275 controlled burns have been carried out, approximately 27 million gallons of oily liquid have been recovered by skimmers, and more than 1.5 million gallons of chemical dispersant have been used in these efforts [169].

In view of the facts that the current oil spill cleaning technology is labor intensive and the toxic chemicals and oil vapors are pernicious to the health of the cleaning crews, there is a pressing need for development and implementation of new technologies for combating oil spills. To mitigate the adverse environmental effects of an oil spill, research efforts are being focused on development of technologies to remove the oil in situ, minimize operational time, and protect health and safety of the cleaning crew [170]. To this end, several novel methods have been developed to make use of autonomous systems for effective oil spill confrontation, such as
Seaswarm [171] (as shown in Fig. 8.2(a)) and Protei [172] (as shown in Fig. 8.2(b)) that are intended to work as a fleet or “swarm” of autonomous systems to create an organized system for autonomous ocean-skimming and oil removal. While the current trend emphasizes hardware improvement, advanced software-based navigation algorithms are yet to be developed.

This chapter develops a multi-resolution method for autonomously cleaning oil spills in dynamic and uncertain environments, as an augmentation of work presented in Chapter 7 in which an autonomous system explores the unknown and static environment and covers the entire search area. The underlying algorithm [163] relies on the notions of both local navigation and global navigation that depend on the spatio-temporal information needed to make these decisions. However, in general, the environment is dynamic due to the spreading and drift of the spills; therefore, the algorithm presented in Chapter 7 is unable to adapt to the weathering process of the oil spill and is thus inadequate for cleaning up the spills. This chapter overcomes this inadequacy by introducing the capability of dynamic adaptation that, upon detection of the oil spills, would enable the autonomous system to replan its actions online. The proposed algorithm is validated on a Player/Stage platform that is capable of high-fidelity simulation of autonomous systems and oil weathering processes for comparison with the benchmark algorithm of back and forth (i.e., zigzag) motion.

The chapter is organized in five sections (including the present section). Section 8.2 presents the modeling of the oil spill process. Section 8.3 provides an
illustrative example of oil spill cleaning in a harbor environment by the proposed algorithm. Section 8.4 summarizes and concludes the chapter with a brief discussion on future work. Appendix D presents a brief background of oil weathering process, as well as its modeling and simulation.

8.2 Modeling of Oil Spill Phenomena and Cleaning

The a priori information, as needed by autonomous systems for oil spill cleaning in dynamic and uncertain environments, is often either incorrect or incomplete. Therefore, time-critical operations of these autonomous systems require real-time decision-making to facilitate continuous adaptation of the evolving information that is generated in situ by onboard sensing and pattern analysis. The generated information refers to the observed phenomena that relate to dynamic unfolding of the search area (e.g., detection of unknown obstacles and boundaries) and environmental changes (e.g., spreading and drift of oil spills). Although such information can be obtained through remote sensing [173], it may not be always available due to communication constraints and high operational cost. Under these circumstances, the autonomous system is required to scan all points in the search area while dynamically discovering new oil spills and avoiding obstacles at a priori unknown locations. This is known as the Complete Coverage Problem [51][143]. A variety of path-planning algorithms exist in technical literature for coverage of the search area using autonomous systems; a review of such algorithms is reported in [144]. Recent coverage algorithms are based on cellular decompositions of the search area at critical points of the Morse functions that correspond to obstacle extremities [53][54].

Although many prototypes of autonomous systems have been developed for oil spill cleaning, navigation algorithms for these prototypes are not adequately addressed [171, 172]. For example, several researchers (e.g., [170, 174, 175]) have tested the algorithms for control of autonomous systems with relatively simple scenarios; however, in the real-world applications, the autonomous systems must carry out the cleanup tasks in more complicated scenarios, such as obstacle-rich
environments that contain islands and other cleaning vessels. This evinces the need for development of navigation algorithms that will enable autonomous systems to perform oil cleaning tasks in dynamic and uncertain environments, where the locations of obstacles and oil spills are \textit{a priori} unknown. From these perspectives, the operation of the oil spill process is modeled under the following assumptions.

1. After occurrence of oil spill at a physical location, the spillage stops before initiation of the cleanup task.

2. The location and volume of oil spill are unknown to the autonomous system under operation but the exact location of the autonomous system is known through a localization system.

3. The autonomous system uses mechanical cleanup to remove the oil at its current position (see Appendix D.1 for details).

4. The dynamic behavior of the oil weathering process can be predicted by an oil transport and weathering model.

The autonomous system, as described in Sec. 6.3.3, is equipped with localization system, navigation sensor, and oil detection sensor that is able to detect oil and other hydrocarbons in water. The remaining part of this section presents the formulation of oil weathering model. The spreading and drift process of the oil spill is modeled by 2D random walk.

\subsection{Oil Spill Modeling}

As stated in Appendix D.1, over 50 oil weathering models have been reported in literature. The 2D random-walk particle-tracking model has been adopted in this chapter because the model is computationally tractable when simulating a large number of particles online and it predicts the time trajectories of the spill size and the probability distribution of the oil spill.

In the random-walk particle-tracking model, spilled oil consists of a large number of particles, with each particle representing a defined quantity of oil. Effectively, model particles are treated as “mass points”, with their transport determined by tidal currents, wind-driven current, turbulent eddies, gravitational
Figure 8.3. Spreading and drift of the oil spill due to wind and current spreading and buoyancy. The 2D update equations [176, 177] for particle positions are given by

\[ X^n = X^{n-1} + A(X^{n-1})\Delta t + B(X^{n-1})Z\sqrt{2K\Delta t} \]  

where \( \Delta t \) is the time interval, \( X^n \) is the position at time \( n\Delta t \) (i.e., at the step number \( n \)), \( A \) is a forcing vector that models the drift process due to currents and wind, \( B \) is a deterministic scaling matrix, \( Z \) is a vector of two independent random numbers taken from a uniform distribution in the range \([-1,1]\), and \( K \) is a vector of the turbulent coefficients. In this model, the motion of one particle is statistically independent of other particles. As seen in Eq. (8.1), the displacement of each particle is determined by its previous position, and the effects of drift and spreading. The effects of other weathering processes (e.g., evaporation, natural dispersion and emulsification) are not included in this model.

The oil weathering process is simulated by following the model in Eq. (8.1), where the simulated scenario is an oil spill incident in a harbor. The oil spillage is assumed to stop before the cleanup process is initiated; however, the spill starts to spread and drift and would eventually hit the port if preventive actions are not taken. Figure 8.3 shows oil particle distribution in the top row and concentration in the bottom row at different time instants, namely, at the beginning of the cleanup (left column) and after 10,000 steps of simulation (i.e. after 3,000 seconds for \( \Delta t = 0.3 \) sec). It is seen in Fig. 8.3 the size of the spill increases due to spreading and the spill moves toward the port due to tidal current and wind. This scenario
is also used in Section 8.3 where autonomous system is dispatched to clean up the oil spill.

**8.2.2 Cleanup Capability**

The multi-resolution algorithm is capable of cleaning up oil spills whose locations may not be known, explained below.

The autonomous system carries out back and forth motion till it detects oil spill. Upon detection of the oil spill, the autonomous system explores the $\kappa_0$-neighborhood of the current grid cell where oil is discovered. Due to the fact that oil distribution is continuous, the autonomous system discovers and cleans up the entire oil slick before exploring the remaining areas. The starting point of the operation is chosen at the windward side such that remaining oil slick does not enter the area that has been cleaned due to drifting. The adaptation neighborhood size $\kappa_0$ is chosen to be large enough to prevent the oil spill from spreading beyond the $\kappa_0$-neighborhood before the autonomous system is able to remove the current oil slick. Following Sec. 6.5.1, the autonomous system eventually finishes covering the entire search area and cleans up all the oil spills. The choice of $\kappa_0$ is a tradeoff between the spreading speed of the oil spill and the operating speed of the autonomous system.

**8.3 Validation on a Simulation Test-Bed**

This section presents validation of the multi-resolution algorithm for oil spill cleaning on a simulation test-bed. The test-bed is built upon the Player/Stage platform that is a high-fidelity open source robotic simulator. The multi-resolution algorithm, together with the 2D random walk oil weathering model, is implemented on the simulator. Several performance metrics are used to comparatively evaluate the performance of the proposed algorithm and the benchmark algorithm in oil spill cleaning. The simulation results of both algorithms are shown in the end of the section.
8.3.1 Player/Stage Simulator

The details of the Player/Stage simulator are given in Sec. 6.6. Although the application of this chapter is in the maritime environment, we still use the Pioneer robot since the navigation algorithm presented in the dissertation is meant to be generic and not platform dependent. The model of the Pioneer robot used in this chapter is the almost same as the one described in Sec. 6.6. The only difference is the Pioneer robot is assumed to equip with oil detection sensor that is implicitly modeled.

The oil spill is modeled as a 2D random walk process. The number of oil particles and their initial locations are specified in the program. The weathering processing of the oil spill is modeled by using a large number of oil particles at every step in the simulation. The navigation algorithm and the oil spill simulation are implemented in separate subroutines in Player such that the autonomous system has no access to the distribution of the oil spill until it enters the grid cell that is occupied by oil particles. A vector is used to keep track of the status of every oil particle. At the beginning of the simulation, the oil particles are at their initial locations and all of them being in the “active” status. The location of each oil particle is updated every step till the autonomous system enter the same grid cell. The oil particles that are in the same grid cell with the autonomous system are labeled to be “inactive”. The “inactive” oil particles are considered to have been removed from the water and thus their locations are no longer updated in the following simulation. This process continues till all oil particles are in the status “inactive”, i.e., the oil spill cleanup task is completed.

In the oil spill simulation, \( N = 5,000 \) oil particles, forcing vector \( \mathbf{A} = [2 \times 10^{-5}, 5 \times 10^{-5}]^T \), scaling matrix \( \mathbf{B} = [2 \times 10^{-2}, 0; 0, 5 \times 10^{-2}]^T \), and vector of turbulent coefficients \( \mathbf{K} = [0.5, 0.5]^T \) are used. The vector \( \mathbf{Z} \) of two independent random numbers is obtained from a uniform distribution with range \([-1, 1]\). Each time step approximately corresponds to \( \Delta t = 0.3 \) sec in real time. Figure 8.3 shows the results of oil spill simulation without any cleaning action. Figure 8.3(a) shows the distribution and concentration of oil spill at the beginning of simulation \( (n = 1) \) and Fig. 8.3(d) shows the same after \( n = 10,000 \) steps of simulation.

A 30m x 30m simulated harbor map has been used to test the performance of
the proposed algorithms. The size of the grid cell at the finest level is \(1m \times 1m\), which is slightly larger than the size of the Pioneer 2AT robot. The selection of the starting point of the operation depends on the direction of tidal current and wind. In this simulation exercise, the starting point is the bottom left corner because the wind and tidal current make the oil spill drift toward the harbor.

Since the initial location of the oil spill is unknown to the autonomous system and both the size and the location of the oil spill change with time due to spreading and drift, the autonomous system needs to cover the entire search area to assure complete cleanup of the oil particles. The typical back and forth motion is optimal for searching an area in terms of minimum number of turns when no adaptation to target and obstacle avoidance is needed. Therefore, for area coverage planning, the exogenous potential field \(B^*_\xi\) in Eq. (6.8) is designed for back and forth motion such that the potential field has a decreasing magnitude from column to column, starting from a maximum value of magnitude 10,000 at the start point, while having equipotential sites on each column. The other parameters in Eqs. (6.3) to (6.9) have been selected to be \(\kappa_0 = 3, \kappa_1 = 2, \phi_T = 0, \phi_U = 0, \phi_O = -40,000, \psi_T = 2,000, \psi_E = 0, T_T = 600, T_U = 1,000\), and \(\alpha = 0.8\) in the simulation exercises; however, the specific values of these parameters do not have very significant effects on the algorithm performance as long as they are of the same relative order of magnitude.

In order to comparatively evaluate the performance of the proposed multi-resolution navigation algorithm, a benchmark algorithm is also tested in the same scenario. In the benchmark algorithm, the autonomous system implements back and forth motion and avoids obstacles as needed. When no unexplored grid cells are in its local neighborhood, the system moves toward the direction of the most unexplored grid cells. In essence, the benchmark algorithm is a simplified version of the multi-resolution navigation algorithm without the adaptation term \(\sum_{<\xi,\nu>_{\alpha}} J_{\xi,\nu} \Psi (\gamma_\xi(t), \gamma_\nu(t))\) as shown in Eq. (6.3). Without the adaptation term, the benchmark algorithm is not affected by the detection of the oil spill in the current grid cell and thus does not deviate from the back and forth motion to search the neighborhood of current grid cell for oil spills.
8.3.2 Performance Metrics

Four different performance metrics are used to compare the effectiveness of the proposed algorithm with that of the benchmark algorithm in oil spill cleaning:

- $T_{\text{total}}$: total time to cover the entire search area
- $T_{\text{clean}}$: time to clean up all the oil spills
- AUC: area under the curve in the oil spill cleaning profile
- EIM: environmental impact measure

The most intuitive performance metric is the completion time. Since the location of the oil spill is unknown and varying with time, the autonomous system needs to cover the entire search area to guarantee removal of all the oil spills. Therefore, two performance metrics regarding the completion time are used in this chapter, namely, the total time in covering the entire search area $T_{\text{total}}$ and the time taken to clean up all oil spills in the search area $T_{\text{clean}}$. It is obvious that, in most cases, $T_{\text{total}} > T_{\text{clean}}$. In the algorithm that has smaller $T_{\text{total}}$, $T_{\text{clean}}$ is considered to be more effective. In the case that these two metrics contradict with each other, the algorithm with smaller $T_{\text{clean}}$ is preferred because the major task in this application is to clean up oil spills in the shortest time.

To keep track of the history of oil spill cleaning, a profile is generated to record the remaining oil spill at each time step. The profiles of the proposed algorithm and the benchmark algorithm are visualized in Fig. 8.6. The metric Area Under the Curve (AUC) has been used to quantitatively compare these two profiles. The formal definition of AUC is given as follows.

**Definition 8.3.1. (AUC)** Let $\mathcal{H}$ be the profile that records the remaining oil spill at each time step, then the metric Area Under the Curve (AUC) is defined as

$$AUC = \sum_{n=1}^{T} \mathcal{H}(n) \quad (8.2)$$

where $T$ is the time taken to complete the task. This metric also emphasizes the efficiency of cleaning the oil spill. The algorithm with smaller AUC is considered to be more effective.
Oil spill has significant ecological impacts to the environment. The longer the oil spill remains in the water, the more damage it will incur. The *Environmental Impact Measure* (EIM) is defined to quantify the impact.

**Definition 8.3.2.** *(EIM)* Let \( \mathcal{H} \) be the profile that records the remaining oil spill at each time step, then the metric Environmental Impact Measure (EIM) is defined as

\[
EIM = \sum_{n=1}^{T} n \mathcal{H}(n) \tag{8.3}
\]

The performance metrics \( T_{\text{clean}} \), AUC and EIM are related to each other. The cleanup time \( T_{\text{clean}} \) shows how fast the algorithm is able to finish cleaning all oil spills, the AUC takes the remaining oil spill at each time step into consideration, and the EIM emphasizes the impact to the environment due to the cumulative effect of the oil spill.

### 8.3.3 Simulation Results

This simulation exercise aims to validate the proposed algorithms for a real-life oil spill cleaning scenario, and compare their performance with the benchmark algorithm. A map with the layout of a typical harbor is designed. The harbor consists of several ports and has one entrance that connects to the open sea, as shown in the top row of Fig. 8.3. The initial oil spill is located at the entrance, probably due to a maritime accident outside the harbor or drifting of the oil spill from the site of an offshore platform accident. As shown in Fig. 8.3, the oil spill will hit the port in certain amount of time due to drift and spreading if no proper response is taken.

The multi-resolution algorithm is used to navigate an autonomous system to clean up the oil spill. Four snapshot are taken and shown in Fig. 8.4(a)-(d). The top row shows the layout of the harbor, the oil spill, and the trajectory of the autonomous system, while the bottom row shows the environment map in the system’s onboard memory. Figure 8.4(a) shows the system follows the offline plan and implements back and forth motion to explore the search area. Figure 8.4(b) shows the system deviates from the offline plan once it detects the oil spill and explores the neighborhood of the grid cells where oil spill is detected. Figure 8.4(c)
shows the moment when the system successfully cleans up all the oil spills in the search area, and Fig. 8.4(d) shows the system explores the remaining search area using back and forth motion and avoids obstacles.

In the environment maps, the yellow box around the search area in Fig. 8.4(a) is the buffer specified at the beginning of the task to prevent the system from leaving the search area. As the exploration goes on, the system detects the jetty and the
Figure 8.6. Oil spill cleaning profiles of the proposed and benchmark methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>$T_{total}$</th>
<th>$T_{clean}$</th>
<th>AUC</th>
<th>EIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmark</td>
<td>9783</td>
<td>3387</td>
<td>2578</td>
<td>$3.51 \times 10^6$</td>
</tr>
<tr>
<td>Proposed</td>
<td>10865</td>
<td>2826</td>
<td>1618</td>
<td>$1.45 \times 10^6$</td>
</tr>
<tr>
<td>Improvement</td>
<td>$-11.1%$</td>
<td>$16.6%$</td>
<td>$37.2%$</td>
<td>$58.7%$</td>
</tr>
</tbody>
</table>

Table 8.1. Comparative Evaluation of the Proposed and Benchmark Methods in Oil Spill Cleaning in a Harbor Environment.

The profiles of oil spill cleaning using the proposed and benchmark methods are shown in Fig. 8.6. The $y$-axis in Fig. 8.6 indicates the normalized number of the remaining oil particles, and the $x$-axis is the time steps in the simulation. It is seen in Fig. 8.6 that the proposed method takes much less time in cleaning up all
the oil spills, although it takes slightly more time to finish scanning the entire the search area. This is understandable since in the proposed method the autonomous system deviates from the offline plan and exploits the neighboring areas when it detects oil spills.

The proposed method and the benchmark method are comparatively evaluated using the performance metrics defined in Section 8.3.2, and the results are shown in Table 8.1. Although it takes 11.1% more time to finish scanning the entire search area, the proposed method is much more efficient in oil spill cleaning and takes 16.6% less time to clean up all the oil spills. As indicated by EIM, the proposed method significantly reduces the impact of the oil spill to the environment by dynamically adapting to the oil spill and replanning online.

8.4 Conclusions and Future work

This chapter presents a multi-resolution navigation algorithm for oil spill cleaning in dynamic and uncertain environments using autonomous systems as a single agent. The concepts of local and global navigation are integrated in the algorithm for adaptive decision-making according to the available spatio-temporal information. The local navigation provides a reduced computational complexity in local decision-making while the global navigation is organized in a hierarchical manner to prevent the robot from being stuck into a local minima. Dynamic adaptation significantly improves the cleaning efficiency and reduces the impact of the oil spill to the environments.

The proposed algorithm has been validated in a harbor example. With the multi-resolution navigation algorithm, the autonomous system manages to explore the complex and uncertain environment, and cleans up all the oil spills in a timely manner. Compared to the benchmark algorithm that uses back and forth motion and obstacle avoidance algorithm, the proposed multi-resolution algorithm is more efficient in oil spill cleaning and significantly reduces the impact of the oil spill to the environment.

While there are many research issues that need to resolved before exploring commercial applications of the proposed algorithms, the following topics are under active research:
• Inference of the oil spill distribution using the collected oil concentration information to facilitate adaptation;

• Validation of the algorithm on hardware platform;

• Extension of the algorithm for multi-agent cooperation.
Summary, Conclusions, and Future Research Directions

The research presented in this dissertation investigates various aspects of autonomous systems. As discussed in the introductory chapter, situation awareness and adaptive decision-making are the key procedures for an autonomous system to sense, plan and act. This dissertation is naturally divided into two parts. Part I deals with sensor data interpretation to extract useful features for classification and fusion. The techniques developed in this part are utilized for intelligent surveillance and reconnaissance (ISR) missions and diagnostic applications. On the other hand, Part II deals with the issue of adaptive decision-making in autonomous navigation. The environment is modeled by a symbolic grid map in the multi-resolution sense for enabling adaptive decision-making using available spatio-temporal information. The navigation algorithm proposed based on this formulation has been shown to be very effective for navigation of autonomous systems in uncertain environments. Integration of the concepts proposed in this dissertation in situation awareness and adaptive decision-making is expected to enable many future applications of autonomous systems, such as fully-autonomous planetary explorer and airborne emergency response to oil spills (AEROS) [178]. The key contributions of this dissertation are delineated in the sequel:
9.1 Contributions of the Dissertation

1. **Wavelet-based feature extraction for time-series data processing**: Class separability information may not be fully reflected in the time domain of a signal due to noise and spurious disturbances. Time-series preprocessing is required to facilitate feature extraction to enhance classification performance. Wavelet transforms of time series have been widely used owing to their time-frequency localization and denoising properties. A wavelet-based feature extraction method is proposed for feature extraction in the two-dimensional scale-shift domain of wavelet transform without any need for non-unique conversion to one-dimensional sequence. The proposed method is validated on the application of behavior recognition in mobile robots.

2. **Optimization of feature extraction via partitioning**: Partitioning is extremely crucial for extracting features under the SDF framework. The dissertation proposes a data partitioning procedure to extract low-dimensional features from time series while optimizing the class separability. A trade-off between sensitivity and robustness of classification is achieved through the construction of the appropriate cost functionals. An application of anomaly detection in nuclear power plants is presented to validate the proposed method.

3. **Multi-level fusion of heterogeneous sensors**: Existing methods of sensor fusion include data fusion, feature fusion and decision fusion. These methods often suffer from drawbacks such as sensitivity to noise and information loss. These shortcomings could be alleviated by the proposed multi-level fusion method in which different levels of fusion schemes are integrated to reduce the information loss and improve the classification performance. Cross machine, a recently developed data fusion tool, is used to extract relational dependencies between different sensor pairs. Feature fusion is then performed to select the most representative features from the ones generated by the cross machine. The proposed algorithm is validated on personnel detection problem where heterogeneous sensors are used to monitor intruders across the borders.

4. **Multi-resolution formulation of the search space for adaptive decision-
**making:** Autonomous systems usually operate in environments with incomplete information and need to make decisions based on the limited information. This dissertation is apparently the first attempt in the research community to formulate the search space in a multi-resolution manner using symbolic grid map to enable adaptive decision-making. A multi-resolution algorithm is proposed for navigation of autonomous systems in uncertain environments. The autonomous systems implement adaptive decision-making based on the available spatio-temporal information about the environments. The proposed algorithm does not suffer from the problem of having local minima, which is commonly encountered in potential-field-based methods.

5. **Autonomous navigation in uncertain environments with complete coverage and dynamic adaptation:** The multi-resolution navigation algorithm is validated on various scenarios that are inspired from real life. The proposed algorithm is able to achieve complete coverage of uncertain and complex environments both effectively and efficiently. Case studies including floor cleaning in an apartment, and exploration in a complex maze and an obstacle-rich environment are presented in the dissertation. The proposed algorithm is also applied to oil spill cleaning in dynamic and uncertain environments. It is shown in the dissertation that the algorithm is more efficient in oil spill cleaning than the benchmark method and significantly reduces the impact of the oil spill to the environment.

### 9.2 Future Research Directions

While future research directions specific to different topics are mentioned at the end of previous chapters, this section presents the broad research areas that can emerge as extensions of the work presented in this dissertation.

1. **Adaptive feature extraction:** Dynamic adaptation to changing environment and signal conditions is the key to situation awareness since theses variations may adversely affect the effectiveness of pattern classification. In the work presented in this dissertation, this problem is circumvented by training
the pattern classifier on the data collected from various conditions. The pitfalls of this method are poor classification performance due to the large variances in the feature vectors and unanticipated scenarios. Adaptive feature extraction that adapts the process of the feature extraction to the changes is more effective than the present method. Adaptation can be made at every stage, i.e., sampling, data preprocessing, feature extraction and classification. In this context, adaptive feature extraction can be performed under the framework of partitioning optimization as introduced in Chapter 4.

2. **Learning of navigation parameters**: In the generalized Ising model presented in Chapter 6, the energy term at a lattice site consists of the adaptation term and the navigation term. The parameters in the interaction function $\Psi$ and the navigation control function $\Phi$ define the behavior of the autonomous system as a trade-off between exploitation and exploration. In this dissertation, the parameters in $\Psi$ and $\Phi$ are determined offline and there is no online update of these parameters. In order to facilitate adaptive decision-making in uncertain environments, a learning algorithm that updates the parameters in $\Psi$ and $\Phi$ and adjusts their relative weights is needed. This can be achieved by tuning the parameters in $\Psi$ and $\Phi$ online based on the feedback from the performance metrics.

3. **Multi-agent system**: The multi-resolution navigation algorithm proposed in this dissertation is designed for autonomous system operating as a single agent. However, the algorithm can be extended to work for multi-agent system (MAS) as well. As introduced in Chapter 6, the probability vector used in global navigation is lightweight and ideal for information exchange between the agents. A possible way to extend the proposed algorithm is to divide the search space and make the autonomous systems work only in their designated task areas. In this case, autonomous systems do not enter the task area of other agents and there is no interaction or cooperation among the agents. This approach inefficient when the complexity of the task areas assigned to the agents is different. It is desirable to make the agents that finish their tasks earlier to assist other agents. A main issue that needs to be addressed here is the coordination among the agents in MAS. This is a
topic under active investigation in the research community.

4. **Seamless integration of situation awareness and adaptive decision-making**: The ultimate goal of many autonomous systems is to achieve complete autonomy. Current advanced autonomous systems such as C4ISR system and Mars rovers still rely on human operators to interpret the obtained information, and set predefined plans or goals. This is due to the inadequacies of either the sensor-based system or the decision-making system. By integrating situation awareness and adaptive decision-making, the autonomous systems will be able to autonomously percept and comprehend the environments, make decisions based on the available information, and take actions to achieve the goals. This will lead to many future applications, such as autonomous landmine removal, swarm robotics for oil spill cleaning, and autonomous Helium-3 mining facility on the Moon.
Appendix A

Principal Component Analysis for Feature Extraction

The time series data sets are organized into an \((M \times N)\)-dimensional data matrix, where \(M\) is the number of data sets, \(N\) is the length of each (one dimensional) time-series data set. Let \(X\) be the centered version of the original \((M \times N)\) data matrix, where each row of \(X\) is an individual data sample \(x\). For \(M < N\), it is numerically efficient [5] to analyze the \((M \times M)\) matrix \(S \triangleq \frac{1}{M}XX^T\) that has the same nonzero eigenvalues as the \((N \times N)\) computed covariance matrix \(\frac{1}{M}X^TX\).

Let \(\{v_i\}\) be the eigenvectors of the matrix \(S\) corresponding to the eigenvalues \(\{\lambda_i\}\) that are arranged in the decreasing order of magnitude. The \(m\) largest (real positive) eigenvalues, where \(m \leq M\), are selected such that \(\sum_{i=1}^{m} \lambda_i > \eta \sum_{i=1}^{M} \lambda_i\), where \(\eta\) is a real positive fraction close to 1. The corresponding normalized eigenvectors \(u_i\) in the original data space are calculated as follows [5].

\[
   u_i = \frac{1}{\sqrt{(M\lambda_i)}}X^Tv_i \tag{A.1}
\]

These \(m\) eigenvectors obtained from Eq. (A.1) are then grouped to form an \((N \times m)\) projection matrix \(W_{PCA}\), where \(W_{PCA} \triangleq [u_1, u_2, ..., u_m]\). Thus, a new low-dimensional data matrix is generated as the \(M \times m\) matrix \(Y_{train} \triangleq X_{train}W_{PCA}\), where the training data set \(X_{train}\) has the same structure as \(X\). Each of the \(m\) rows of \(Y_{train}\) is considered as a feature vector. For subsequent pattern classification task, a reference pattern \(\bar{p} = \frac{1}{M}1X_{train}\) is generated for each class,
where $\mathbf{1} \triangleq [1 1 \ldots 1]$.

Following the above procedure, we obtain the projection matrix $W_{PCA}$ and the low-dimensional data matrix $Y_{train}$ from the training data matrix $X_{train}$. Similarly, in the testing stage, the low-dimensional data matrix is obtained as $Y_{test} = X_{test} W_{PCA}$, where each row of $Y_{test}$ is a feature vector $p$ that corresponds to a test sample $x$.

In summary, to apply PCA to high-dimensional data, we first evaluate $XX^T$ and then find its eigenvectors and eigenvalues and then compute the eigenvectors in the original data space [5].
Appendix B

Description of the IRIS Test-Bed

The International Reactor Innovative & Secure (IRIS) nuclear power plants is based on the design of a next-generation nuclear reactor. It is a modular pressurized water reactor (PWR) with an integral configuration of all primary system components. Figure B.1 shows the layout of the primary side of the IRIS system that is offered in configurations of single or multiple modules, each having a power rating of 1000 MWt (about 335 MWe) [179]. The nominal reactor core inlet and outlet temperatures are 557.6°F (292°C) and 626°F (330°C), respectively. The pressurizer, eight steam generators, and the control rod mechanism are integrated into the pressure vessel with the reactor core. There is no huge pipe used to connect these components. This design avoids the large loss of coolant accident (LOCA). The entire control rod mechanism is mounted inside the pressure vessel to avoid the problem of penetrating the pressure vessel head by the control rod.

In order to develop an automated anomaly detection algorithm, it is necessary to “teach” or “train” a pattern classifier so that it captures the process dynamics and produces the desired output for the given inputs. For systems where this cannot be done experimentally, high-fidelity simulators should be constructed for developing anomaly detection algorithms. The IRIS simulator, used in this dissertation, has been built by researchers at North Carolina State University [180, 181, 182]. In the normal operating range, reactor power is determined through a point kinetics model, with rod position controlled through a user-specified program. Feedwater control is based upon steam demand, similar to control strategies employed in B&W once-through steam generators [183]. In the
very low power range, a constant heat input is assumed to simulate decay heat, and the average moderator temperature is allowed to float.

The test-bed is built using FORTRAN programming language. This FORTRAN model includes a reactor core model, a helical coil steam generator (HCSG) model. The test-bed is implemented on a Quad Core 2.83 GHz CPU 8 GB RAM Workstation in the laboratory of Penn State. The IRIS simulator is operated in the integrated control mode through built-in PID controllers, which operates all three subsystems (i.e., turbine, feedwater flow and control rods) to combine the rapid response of a reactor-following system with the stability of a turbine-following system [183].
The IRIS test-bed is capable of simulating both normal conditions at different operational modes (e.g., normal power maneuvers, reactor start-up, and turbine loading) [181]) and a variety of anomalous scenarios that include:

- Actuator anomalies, e.g., feedwater pump trip, malfunctions of reactor coolant pump and control rod mechanism;
- Sensor failures, e.g., malfunctions of temperature, pressure, and flow-rate sensors;
- Internal anomalies, e.g., uncertainties in the fuel temperature coefficient of reactivity, coolant heat capacity, and feedwater heat capacity.

In the IRIS test-bed, sensor degradations are realized as injected noise and disturbances. Depending on the location and modality of a sensor, there could be several different degradation levels. For example, the degradation levels in a pressure sensor have different characteristics from those of a temperature sensor. Furthermore, depending on the location and operating environment, even sensors of the same modality could have different degradation characteristics. In general, sensor degradation is categorized as the following [184]:

- Constant bias and drift (i.e., slowly-varying bias);
- Change in sensor response time due to aging; and
- Change in the variance of sensor noise (e.g., due to large external electromagnetic radiation from electric motors).

Amongst the above sensor degradation types, only sensor degradation due to changes in the noise variance are investigated in this dissertation. The rationale is that the sensors are assumed to be periodically tested and calibrated; hence, sensor degradation due to aging, bias, and drift is much less likely.
Information Fusion

C.1 Levels of Information Fusion

There are different levels of information fusion. We can have information fusion from one sensor (time series), redundant sensors, redundant variables and systems [185]. Different approaches can be found in the literature to tackle this problem, such as statistical analysis [186], heuristic methods [187], probability models (fuzzy logic and Dempster-Shafer theory [188]), mathematical models [189], learning algorithms and hybrid systems [190]. In general, information fusion processes are often categorized in three levels of modes: data fusion, feature fusion, and decision fusion [191].

C.1.1 Data Fusion

Low-level fusion or data fusion, as shown in Figure C.1, combines several sources of essentially the same type of raw preprocessed data to produce a new data set that is expected to be more informative and useful than the inputs. This is usually done when the sensor types are competitive. The fusion of competitive type of sensors would provide robustness and fault-tolerance because comparison with another competitive sensor can be used to detect faults. For example, in the case of face biometrics, both 2D texture information and 3D depth (range) information (obtained using two different sensors) may be fused to generate a 3D texture image of the face which could then be subjected to feature extraction and matching [192].
C.1.2 Feature Fusion

Intermediate-level, or feature-level fusion, as shown in Figure C.2, combines various features such as edges, lines, corners, textures, or positions into a feature map. This map is used for segmentation of images, detection of objects, etc. This process of fusion is called feature-, or image-level fusion.

Feature level fusion is usually used to determine or verify the identity of an individual from multiple biometric sources. The feature sets extracted from multiple data sources can be fused to create a new feature set to represent the individual. The geometric features of the hand, for example, may be augmented with the eigen-coefficients of the face in order to construct a new high-dimension feature vector. A feature selection/transformation procedure may be adopted to elicit a minimal feature set from the high-dimensional feature vector [192].
**C.1.3 Decision Fusion**

High-level fusion, or decision fusion, as shown in Figure C.3, combines decisions from several experts/classifiers. Methods of decision fusion are voting, fuzzy logic, and statistical methods.

In many machine learning problems, combining the decisions of several classifiers has shown to be an effective technique for improving the classification performances. Dietterich [193] gives three main reasons why an ensemble of classifiers might be a better choice than a single classifier. First, when the same classification accuracy can be achieved by several classifiers (in particular when the available training set is small), averaging all the decisions can reduce the risk of picking the wrong classifier. Then, many learning techniques use local searches to converge toward a solution, with the risk of staying stacked in local optima. Running several searches and combining the solutions can improve the performance. Finally, from a representational perspective, we may not be able to obtain the optimal classifier using a given training set and a given classifier architecture. Combining several classifiers can produce a better approximation of the optimal solution.

A common theoretical framework for combining classifiers which use distinct pattern representations was developed in [194], which examined various classifier combination schemes and claimed the sum rule outperformed other classifier combination scheme. In [195], an information theoretic framework was proposed to establish a link between the diversity between classifiers and the individual accuracies of the classifiers. The trade-off between individual accuracy and diversity was expressed by information theoretic score.
C.2 Architectures of Sensor-Fusion Networks

Sensor-fusion networks are organized topologies that have specified architectures and are categorized according to the type of sensor configuration, as described below [31]:

**Complementary type** In this type of sensor configuration, the sensors do not depend on each other directly. One sensor views part of the region, and another views a different part of the region, thereby giving a complete picture of the entire region. Because they are complementary, they can be combined to establish a more complete picture of the phenomenon being observed and hence the sensor datasets would be complete. Fusion of complementary data is relatively easy because the data from independent sensors can be appended to each other.

**Competitive type** In this type of configuration, each sensor delivers independent measurement of the same attribute or feature. Fusion of the same type of data from different sensors or the fusion of measurement from a single sensor obtained at different instants is possible. This configuration would provide robustness and fault-tolerance because comparison with another competitive sensor can be used to detect faults. Such robust systems can provide a degraded level of service in the presence of fault; moreover, the competing sensors in this system do not necessarily have to be identical.

**Cooperative type** In this type of configuration, data provided by two independent sensors are used to derive information that would not be available from a single sensor, as in the stereoscopic vision system. Cooperative sensor fusion is difficult to designing, and the resulting data will be sensitive to the inaccuracies in all the individual sensors.
Background of Oil Spill

This appendix briefly reviews the background of oil spill. The relevant information regarding the oil weathering process and the oil spill cleaning methods are discussed.

D.1 Oil Weathering Processes

The spilled oil is normally a mixture of hydrocarbon compounds whose chemical and physical properties vary among oil types. When an oil spill occurs, the nature of the oil undergoes a series of changes in chemical and physical properties over time, that, in combination, is termed “weathering”. The oil weathering processes include short-term processes such as spreading, drift, evaporation, emulsification, dispersion, and dissolution, and long-term processes such as photooxidation, biodegradation, and sedimentation [169, 196].

In the event of a massive oil spill on water, the oil begins to spread by gravity, wind, and current, with the process resisted by inertia, viscosity, and surface tension, until the slick reaches a thickness of 0.1mm or less. The process, known as spreading, can affect other weathering processes such as evaporation, dispersion, and emulsification. The environmental impact of oil spills largely depend on the spreading area. The cleanup operations also require information on the spill size. Because of the influence of the winds and wind-induced surface currents, the oil slick may move downward with respect to the wind direction. This movement, called drift, results in a displacement of the center of the oil slick and contributes
to the non-symmetrical spreading [169]. This dissertation considers both spreading and drift in the oil weathering process. Other weathering processes include evaporation, natural dispersion, emulsification, dissolution, photo-oxidation, sedimentation, and biodegradation.

In the oil weathering process, a variety of complex physical, chemical and biological phenomena take place simultaneously. The weathering process depends on the initial oil properties, the spilled amount, hydrodynamics, and weather conditions [169]. It is estimated that over 50 oil weathering models have been developed. The transport of oil can be modeled either by solving the advection-diffusion equation on the node of a grid (Eulerian models) [197, 198], or by tracking particles which represent individual portions of the spill (Lagrangian models) [199]. Random walk models [200, 201, 176, 177], which solve nonlinear Langevin equation, belong to the later category. For the 2D random walk models, initial conditions including the number of particles and time interval need to be specified. For random walk spill models involving the vertical direction, oil particle size distribution must also be specified, since oil buoyancy influences oil spill movement and spreading [200]. In this dissertation, the 2D random-walk particle-tracking technique [176, 177] is used to follow the motion of individual particles (oil droplets), the total amount of which constitutes the oil spill.

D.2 Oil Spill Cleanup Methods

When a massive oil spill occurs, quick and effective response is critical in order to minimize the economic impact and the damage to both the ecology and the quality of human life. Five methods are commonly used to contain and clean up a spill: mechanical methods, chemical methods, to the use of absorbents, oil burning, and bioremediation [202].

The current state of the art in spill countermeasures varies from mechanical and chemical methods, to the usage of absorbents, oil burning and bioremediation [202]. Mechanical methods, referring to the use of booms and skimmers, are the most widely used combat means. Booms are used to confine the oil to a specific area, hence controlling its spreading, and/or stop the oil from entering a given area, while skimmers are used to recover the oil from the water surface [203]. Other cleanup
methods may be effective under certain circumstances, but they also suffer from many limitations and are not as cost effective as the mechanical methods [170].

The chemical methods are more efficient than mechanical method when properly applied, but the dispersed oil droplets in high concentrations could have an acute lethal toxicity for many species. Despite burning of the spilled oil is an efficient, versatile and lower cost method, there could be possibility of sinking extremely viscous and dense residues. The idea of bioremediation is attractive, however, its practical use is restricted.

Mechanical cleanup refers to the use of booms and skimmers, and are the most widely used combatting means. Mechanical recovery operations for large spills are in general considered to be expensive, complex and labor-intensive with their recovery efficiency not to usually exceed 20% [204]. The chemical methods are based on the transformation of the physicochemical properties of the oil, and the most common method is the use of dispersants. Although dispersants, when properly applied, are more efficient than skimmers, for large spills can also be expensive, complex and labor-intensive. In situ burning corresponds to controlled ignition and burning of the oil at or near the spill site, on the surface of the water or in a marsh [204]. Despite burning of the spilled oil is an efficient, versatile and lower cost method, there could be possibility of sinking extremely viscous and dense residues. Bioremediation refers to the addition of microorganisms that are able to degrade hydrocarbons, or the addition of fertilizers to the hydrocarbon degrading bacteria and fungi that naturally exist in marine environments. Although the idea of bioremediation is attractive, its practical use is restricted.

Current technology to skim oil spill harms the health of cleaners, cannot operate at night safely, is not in operation far from shore because it is too costly, and cannot operate in rough weather.
Bibliography


Vita

Xin Jin

Xin Jin was born in Yiwu, Zhejiang, China on July 29, 1985. He received the B.S. degree in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in July 2007. In his senior year of undergraduate study, he attended the Global Engineering Alliance for Research and Education (GEARE) program at Purdue University as an exchange student.

Xin Jin started his graduate studies at Penn State in August 2007, and received the M.S. degrees in both electrical engineering and mechanical engineering from Pennsylvania State University, University Park, where he is currently pursuing the Ph.D. degree in mechanical engineering. He also worked at Bosch Research and Technology Center in Pittsburgh, PA as a research intern in the area of residential energy management between May 2011 and December 2011.

Xin Jin joined Dr. Ray’s group in June 2008 and is active in interdisciplinary research. His current research interests include machine learning, signal processing, instrumentation & control, energy management, and robotics. At Penn State, he successfully participated in multi-disciplinary teams for multiple projects sponsored by ARO, ONR, DOE and NASA. He has co-authored ten journal papers and more than fifteen refereed conference papers.

Xin Jin received a Best Paper in Session Award in 2010 American Control Conference at Baltimore, Maryland. He has been a student member of the Institute of Electrical and Electronics Engineers (IEEE) and the American Society of Mechanical Engineers (ASME) since 2009. He has also served as a reviewer for more than ten technical journals and conferences.

Xin Jin plans to join the National Renewable Energy Laboratory (NREL) at Golden, Colorado as a postdoctoral researcher in September 2012.