ROBOT SHARE: A FRAMEWORK FOR ROBOT KNOWLEDGE SHARING

by

Xiuyi Fan

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computational Engineering and Science

School of Computing

The University of Utah

May 2009

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a thesis submitted by

Xiuyi Fan

This thesis has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the thesis of Xiuyi Fan in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Thomas C. Henderson
Chair: Supervisory Committee

Approved for the Major Department

Martin Berzins
Chair/Director

Approved for the Graduate Council

David S.
Dean of The Graduate School

ABSTRACT

Knowledge representation is a traditional field in artificial intelligence. Researchers have developed various ways to represent and share information among intelligent agents. Agents that share resources, data, information, and knowledge perform better than agents working alone. However, previous research also reveals that sharing knowledge among a large number of entities in an open environment is a problem yet to be solved. Intelligent robots are designed and produced by different manufacturers. They have various physical attributes and employ different knowledge representations. Therefore, any nonstandard or non-widely-adopted technology is unsuitable to provide a satisfactory solution to the knowledge sharing problem. In this research, we pose robot knowledge sharing as an activity to be developed in an open environment - the World Wide Web. Just as search engines like Google provide enormous power for information exchange and sharing for humans, we believe a searching mechanism designed for intelligent agents can provide a robust approach for sharing knowledge among robots. We have developed (1) a knowledge representation for robots that allows Internet access, (2) a knowledge organization and search indexing engine, and (3) a query/reply mechanism between robots and the search engine.

CONTENTS

ABSTRACT iv

LIST OF FIGURES vii

LIST OF TABLES ix

CHAPTERS

1. INTRODUCTION 1
   - 1.1 Motivation 1
   - 1.2 Introduction 2
   - 1.3 Relevant Work 5

2. KNOWLEDGE REPRESENTATION 9
   - 2.1 Robot Knowledge 9
     - 2.1.1 Sensor Grounded Knowledge 9
     - 2.1.2 Knowledge Definition 10
   - 2.2 Knowledge Extraction 12
     - 2.2.1 Data Type and Extraction 12
     - 2.2.2 Data Sample 13
   - 2.3 Knowledge Formulation 15
     - 2.3.1 The First Transformation 18
     - 2.3.2 The Second Transformation 19
     - 2.3.3 Example 24
   - 2.4 Activity Knowledge 25

3. KNOWLEDGE SEARCH ENGINE 28
   - 3.1 Robot Knowledge Search Engine 28
   - 3.2 Knowledge Harvesting 29
   - 3.3 Knowledge Query 29
     - 3.3.1 Query Type 30
     - 3.3.2 Query Interface 30
     - 3.3.3 Search Type 31
   - 3.4 Similarity Functions 31
     - 3.4.1 Distance Measures 31
     - 3.4.2 Experiment Data 34
   - 3.5 Feature Indexing 35
     - 3.5.1 Curse of Dimensionality 35
     - 3.5.2 Feature Selection 36

3.5.3 Dimensionality Reduction 41
3.5.4 Indexing Structures 42
3.5.5 Indexing Text Data 44
3.6 Robot Share Architecture 46
3.6.1 Indexing Structure Consideration 46
3.6.2 Robot Share Component 50

4. KNOWLEDGE EXPLOITATION 56

4.1 Object Knowledge Experiment 56
4.1.1 General Performance 57
4.1.2 Query with Missing Fields 58
4.1.3 Ranking 64
4.1.4 Supplemental Experiment I 71
4.1.5 Supplemental Experiment II 73

5. FUTURE WORK AND CONCLUSION 74

5.1 Robot Share Refinement 74
5.2 Robot Share Expansion 78

APPENDICES

A. DIMENSIONALITY REDUCTION 81

B. THE KNOWLEDGE DEFINITION GRAMMAR 102

C. THE QUERY RESPONSE GRAMMAR 105

REFERENCES 106

LIST OF FIGURES

1.1 The Robot Share Framework. 3

2.1 An Image Retrieval System Architecture. 14

2.2 A Sample Image. 15

2.3 A Sample Color Histogram. 16

2.4 A Sample Edge Histogram. 17

2.5 An Edge Histogram and its Polynomial Approximation. 24

3.1 Distances Between a Query Point and Database Samples. Sample Size = 20,000; Vector Dimension = 100; Bin Number = 100. D_{min} = 10.22, D_{max} = 17.41, D_{avg} = 13.94 37

3.2 Distances Between a Query Point and Database Samples. Sample Size = 20,000; Vector Dimension = 1; Bin Number = 100. D_{min} = 0, D_{max} = 5.16, D_{avg} = 1.11 38

3.3 Distances Between a Query Point and Database Samples. Sample Size = 20,000; Vector Dimension = 5; Bin Number = 100. D_{min} = 0.28, D_{max} = 6.05, D_{avg} = 2.72 39

3.4 The Single-indexing-structure Architecture. 46

3.5 The Multi-indexing-structure Architecture. 47

3.6 K-d Tree Structure of Two Bowls and Two Knives. 50

A.1 Confusion Matrix of the Edge Direction Histogram in Kullback-Leibler Divergence 82

A.2 Confusion Matrix of the Edge Direction Histogram in L1 Distance . . . 83

A.3 Confusion Matrix of the Edge Direction Histogram in L2 Distance . . . 84

A.4 Confusion Matrix of the Edge Direction Histogram in Weighted L1 Distance 85

A.5 Confusion Matrix of the Edge Direction Histogram Represented by Fourier Coefficient in Kullback-Leibler Divergence 86

A.6 Confusion Matrix of the Edge Direction Histogram Represented by Fourier Coefficient in L1 Distance 87

A.7 Confusion Matrix of the Edge Direction Histogram Represented by Fourier Coefficient in L2 Distance 88

A.8 Confusion Matrix of the Edge Direction Histogram Represented by Fourier Coefficient in Weighted L1 Distance 89
A.9 Confusion Matrix of the Edge Direction Histogram Represented by Coefficients of a Polynomial in Kullback-Leibler Divergence 90
A.10 Confusion Matrix of the Edge Direction Histogram Represented by Coefficients of a Polynomial in L1 Distance 91
A.11 Confusion Matrix of the Edge Direction Histogram Represented by Coefficients of a Polynomial in L2 Distance 92
A.12 Confusion Matrix of the Edge Direction Histogram Represented by Coefficients of a Polynomial in Weighted L1 Distance 93
A.13 Confusion Matrix of the Edge Direction Histogram Represented by Statistical Properties in Kullback-Leibler Divergence 94
A.14 Confusion Matrix of the Edge Direction Histogram Represented by Statistical Properties in L1 Distance 95
A.15 Confusion Matrix of the Edge Direction Histogram Represented by Statistical Properties in L2 Distance 96
A.16 Confusion Matrix of the Edge Direction Histogram Represented by Statistical Properties in Weighted L1 Distance 97
A.17 Confusion Matrix of the Edge Direction Histogram in Represented by Central Moments Kullback-Leibler Divergence 98
A.18 Confusion Matrix of the Edge Direction Histogram Represented by Central Moments in L1 Distance 99
A.19 Confusion Matrix of the Edge Direction Histogram Represented by Central Moments in L2 Distance 100
A.20 Confusion Matrix of the Edge Direction Histogram Represented by Central Moments in Weighted L1 Distance. 101

LIST OF TABLES

2.1 A Sample Object. 26
3.1 The Robot Share Component Length Summary. 52
4.1 Performance Test 1.1a 58
4.2 Performance Test 1.1b 58
4.3 A Query With Only One Image and No Weight Measure. 59
4.4 Performance Test 2.1 59
4.5 Performance Test 2.2 60
4.6 Performance Test 2.3 60
4.7 Performance Test 2.4 61
4.8 Performance Test 2.5 61
4.9 Performance Test 2.6 62
4.10 Performance Test 2.7 63
4.11 Performance Test 2.8 63
4.12 Performance Test 2.9 64
4.13 Performance Test 3.1 Part I 65
4.14 Performance Test 3.1 Part II 66
4.15 Performance Test 3.1 Part III 67
4.16 Performance Test 3.2 69
4.17 Performance Test 3.3 71
4.18 Supplemental Experiment I 72
4.19 Supplemental Experiment II 73

CHAPTER 1

INTRODUCTION

1.1 Motivation

Knowledge representation is a traditional field of study in artificial intelligence [34, 37]. More recently, knowledge sharing has attracted research interest. Previous research has been focused on the formation of knowledge, representation of knowledge, categorization and partition of knowledge, etc. Various knowledge base structures, knowledge interchange languages, and knowledge sharing infrastructures have been developed. Knowledge sharing among intelligent agents, e.g., robots, brings more power to each participating agent as it can accomplish its jobs more rapidly and/or at a lower cost.

One problem of the previous studies was that most dealt with a closed environment and defined a specific set of knowledge representation structures and communication languages in a somewhat nonstandard fashion. This limits the adoptability and the flexibility of those systems. Another problem is the scalability of these systems, as most of them are designed to operate among a small number of participants. These systems may work well in a small LAN, but when they are adapted to an open WAN, various problems emerge, i.e., network latency, platform incompatibility, communication language incompatibility, etc. One major goal is to vastly increase the scale of robot knowledge sharing.

We build a knowledge sharing framework that supports a large number of data formats, is able to scale to process large amounts of data and is accessable to a large number of robots. To avoid some of the problems of earlier approaches, we develop a web-based approach for knowledge sharing among robots.

Thesis: Robots sharing knowledge is more efficient and successful than robots learning and acting on their own. A web-based open archi-

tecture helps to bring more robots into the system and enhances their performance.

1.2 Introduction

In the past, when we needed to know something, we would look it up in an encyclopedia or find a book on the subject. Nowadays, we turn to web search engines, such as Google^{TM}[1] or Yahoo^{TM}[2], and are given pointers to a large amount of information. We usually find what we are looking for relatively quickly and easily. The semantic web holds promise for the future in which communities of practice will share knowledge to meet their needs or solve problems. We propose to develop similar capabilities for physical robots, including humanoid robots, which act in the world and must know a great deal about it. Humanoid robots in our research include robot butlers, surgeons, drivers, hospital orderlies, homecare nurses, etc. Thus, when a robot encounters an unfamiliar or unknown object in its environment, or when it needs to know how to perform a particular task with or on an object (e.g., clean it), it will be able to query a Robot Google^{TM} in order to get pointers to relevant information available in the world wide web, or it will interact with a robot knowledge ontology-based sharing community.

Humans achieve this sharing mainly through natural language: queries are words that are matched to document content. For robots, it is not clear how to achieve this, and the question arises as to what representations best facilitate robot knowledge sharing. Restricting for the moment our consideration to 3-D physical objects, a description may include geometry, physical properties, functional use, context, and natural language descriptions. Other knowledge, e.g., task procedures, may require representation of desired forces, torques, wrenches, etc., described in an appropriate sharable representation (e.g., some form of configuration space). The

[1] GOOGLE is a trademark of Google Inc.

[2] Yahoo is a trademark of Yahoo Inc.

development of and access to networked robot knowledge can provide the basis for very robust intelligence for robot systems.

The developed framework for our solution is shown in Figure 1.1. In this figure, each participant robot creates web accessible knowledge repositories. the Robot Share server harvests knowledge from each of the participant robots and then organizes and creates efficient indexes into the database. Participant robots query the Robot Share server for knowledge and receive URLs pointing to other robots' knowledge. Robots, in this view, act as agents [12, 29, 39, 40, 43], and we assume their ability to generate the necessary knowledge structures, this is not an issue of investigation here.

Imagine a scenario like the following. A kitchen robot works in a kitchen. It is told to clean all kitchenware. After successfully cleaning a few plates, forks and spoons, the robot notices that there is a pair of wooden sticks. The robot is confused by this pair of wooden sticks as it has never seen this before. The robot does not know what to do with them. By asking a nearby human, the robot learns that this

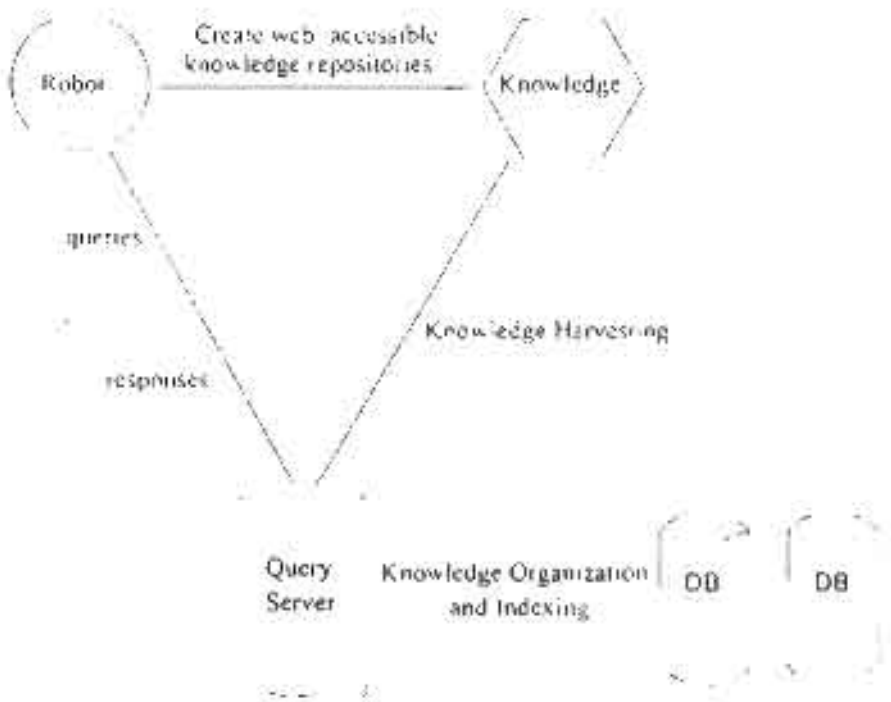

Figure 1.1 The Robot Share Framework

pair of wooden sticks is called "chopsticks." The robot then formulates a query and sends it to the Robot Share server through its on-board Internet connection. It wants to know if these chopsticks need to be cleaned (as it is told to clean all kitchenware) and if so, how to clean them. The Robot Share server processes the query, and responds with: please go to www.robot_chopstick.com/info to see more information about chopsticks and please go to www.robot_chopstick.com/clean to see how they can be cleaned. The robot then connects to those two provided URLs and downloads a few packages that contain the needed information. It then cleans the pair of chopsticks successfully.

From the scenario described above, we can see that there are at least two types of knowledge a robot may be interested to know,

- The knowledge of object identification, i.e., the knowledge to answer questions like: what is this?
- The knowledge of object manipulation, i.e., the knowledge to answer questions like: what can be done to/with this object?

Both of the two types of knowledge are closely tied to a robot's physical capacities, i.e., a robot's sensors and its actuators, as sensor data are utimately what a robot knows about the external world and actuators define what a robot can possibly do to the external world. Therefore, the proposed framework emphasizes sensor information. We believe sensor data provide a solid grounding for this research.

Goals for this research are:

- develop robot knowledge sharing framework,
- study robot knowledge representation,
- study knowledge indexing structure,
- study accuracy of retrieved knowledge,
- study performance of sharing.

1.3 Relevant Work

The study of knowledge representation can be traced back to ancient Greece. Epistemology, the study of the nature of knowledge and its justification, was established by Plato in the fifth century B.C. [37]. Since then, the study of knowledge, including its nature, representation, development, etc., has been carried on by philosophers, mathematicians, linguists, and scientists. Most knowledge representation developed today is rooted in various logics. Recently, some computer scientists have expressed belief that grounding knowledge purely in logic, e.g., in symbolic languages, is insufficient for building intelligent agents and robots. They propose to develop sensor grounded and context-aware knowledge representations for robots [32, 33]. Even though their work is promising, they are still far from providing a comprehensive and satisfactory solution.

Although still in its formative stages, several groups are making progress on sensor-grounded robot knowledge creation. In our provisioning effort we intend to take advantage of this. Cohen et al. [6] describe a *natural semantics* approach in which robots learn meanings through their interaction with the environment. Traditional AI approaches rely on the reduction of semantics to syntax, and such systems have no real understanding of the symbols that they manipulate. In natural semantics, such meanings are acquired and maintained by the robot system, and not specified externally by human programmers or knowledge engineers. In this work, a robot is provided with a small number of behaviors (e.g., move, turn, open gripper, etc.), and the robot records sensor data streams. From this, prototype sequences are segmented and serve as the basis for more complex tasks. In this way, the robot learns a sensor data based ontology through interaction with the environment, and concepts are related to the sense data.

Another approach is the Spatial Semantic Hierarchy which allows bootstrap learning from uninterpreted experience. This involves solving three problems: (1) *feature learning* from the sense data, (2) *control learning* for achieving desired states, and (3) *place recognition* to identify distinctive states. Ref. [26] explains this approach in detail.

> Starting with completely uninterpreted sense and motor vectors, as well as an unknown environment, we show how a learning agent can separate the sense vector into modalities, learn the structure of individual modalities, learn natural primitives for the motor system, identify reliable relations between primitive actions and created sensory features, and can define useful control laws for homing and path following.

This fits well with our robot knowledge provisioning scheme since raw data, as well as learned structures, will be available.

Grupen et al. have based their approach on human developmental theory [18]. They have demonstrated a framework for the development of robot behavior in which:

> All behavior is initially constructed from a set of innate control laws and events that delineate control decisions and are derived from the pattern of (dis)equilibria on a working subset of sensorimotor policies. We show how this architecture can be used to accomplish sequential knowledge gathering and representation tasks and provide examples of developmental learning using a quadrupedal walking robot.

In addition, they have proposed a relational representation for procedural task knowledge [19]. Joint probability estimates are learned which relate features of the sensorimotor stream to desired behavior quality. In this way, the robot can determine salient features in its world experiences (sensor/actuator mediated) and choose action policies. This group has examined many issues related to human-like activity (e.g., grasping, walking, etc.).

As a last example of a group producing sharable robot knowledge (there are many more; we have selected a representative sample briefly), DiLmann et al. have focused on robot knowledge related to their humanoid project [3]. Their recent PACO-PLUS project aims to develop a cognitive robot [10]:

> [Our approach is] capable of developing perceptual, behavioural and cognitive categories in a measurable way and of communicating and sharing these with humans and other artificial agents. To achieve this, the project brings together a consortium of roboticists, researchers, engineers, computer vision scientists, linguists, theoretical neuro-scientists and cognitive psychologists. The systems we aim at are supposed to interact and function together with humans. They are meant to be able to cooperate and to enter a dialogue communicating with the human. Therefore they need to understand both, what they perceive and what

> they do. Our hypothesis is that such understanding can only be attained if we consider perception and action together. In this process the artificial system needs to learn and adapt to the momentarily existing situation to be able to act and react appropriately.

Their work will provide a way to bridge knowledge between humans and robots. They have recently proposed a reference model for human kinetics just for the purpose of enabling sharing [2].

In order to exchange knowledge, robot agents also require a common language for the expression of their data and processes. As a starting point, common sensors and actuators give a direct mechanism for exchange. Analysis of the sensor data is then straightforward, as well as control of actuators. More abstract sharing mechanisms are possible when specific sensors and/or actuators differ. For example, Logical Sensor Systems [9, 20] provide such a framework. In this case, sensors are abstracted as a data type in an object-oriented sense. Physical operations by the agent on the world may be expressed as a sequence of force closures to be achieved (e.g., in terms of forces, torques, wrenches, etc.) [21].

Another influential work in knowledge sharing is the Knowledge Interchange Format, known as KIF [17]. KIF was defined as an ANSI standard by the NCITS T2 committee on Information Interchange and Interpretation in 1998. KIF is a version of typed predicate logic. It is still unclear what is the most appropriate knowledge representation format for robots, and exchanging knowledge between robots in an unrestricted environment is still a problem to be solved.

We are aware the current research on the Semantic Web [7], which is led by the World Wide Web Consortium. The aim of the Semantic Web project is to create a universal medium for information exchange by putting documents with computer processable meaning on the World Wide Web. Using the Semantic Web, information can be better organized and more accurately delivered to a human reader. The book by Davies et al. [7] provides a very clear review of methods and tools developed for the human semantic web, including methods to extract information from text, retrieve information from other sources, and to compress, visualize and disseminate information.

We would like to position Robot Share to a starting point for sharing data and information among robots. How to reformat the structure of Semantic Web to suit robots will be an interesting topic for furture research. It would also be an exciting and difficult task to introduce a mathematical theory to model this problem.

Chapter 2 presents our view of robot knowledge. It introduces the knowledge representation we have developed. Chapter 3 presents various issues involved in a search engine construction and introduces the architecture of the Robot Share search engine. Chapter 4 presents the performance of Robot Share. Chapter 5 gives an overview of possible future research and conclude this thesis.

CHAPTER 2

KNOWLEDGE REPRESENTATION

2.1 Robot Knowledge

2.1.1 Sensor Grounded Knowledge

Humans recognize the external world first through sensory organs. When we visit a museum, we see a number of artifacts on display. Suppose there is an object we do not recognize, and we would like to know what it is. We look at it to see its shape; we lift it (assuming it is permitted to do so) to feel its weight; we may smell it, even though it may not be very helpful for this case, to determine its odor; we may tap it with our finger to see how it sounds (again, assume it is permitted to do so). With this collected sensory information, we try to associate this new object to some object we already know. Association is a pattern matching process, i.e., we try to find a known object, which has its sensory property close to the new object, in our memory. If we cannot find such an object, or we find multiple ones and we are not so sure which one is the closest, we may check out the museum description card, as it provides some verbal information that may help us to distinguish the best match. We may also look around to see where this object is placed, as knowing the object's environment may help us to recognize the object. To summarize, we recognize an object first by collecting information through our sensory organs. This information comes in various formats. Some of them are more accurate than others; some of them are more abstract than others; some of them have temporal properties while others do not. It seems there is enough evidence for us to believe that sensory information, i.e., information collected by our sensory organs, is the ground for all of our object recognition process.

We believe robots can behave similarly and that the best way for a robot to recognize objects is through sensor data. We would not deny ontology information

like *a fork is a kitchenware; a kitchenware is a tool used for dining or relevant purpose; a tool used for dining or relevant purpose usually is an artifact; an artifact is an object* could be helpful at some place in our intellectual system. However, we believe such information is less consulted, if at all, when we encounter something previously unknown. Therefore, we would not deny ontology or logic, at least to some extent, could be helpful for a robots. However, we would like to focus this research towards a sensor data grounded approach.

We are aware that currently there is a good deal of research dedicated to various aspects of robot vision [5, 27] to find methods for object segmentation, object tracking, etc. Albeit interesting, these topics are beyond the scope of this research. We assume robots are able to identify objects in the environment (i.e., segment them in senser data) and measure physical properties of the object using on-board sensors.

2.1.2 Knowledge Definition

The definition of knowledge is still fuzzy at this point, as philosophers love to debate this type of topics. The classic definition, found in Plato, states that three criteria define knowledge: knowledge needs to be a statement, such that it is justified, true, and believed [42]. For the purpose of this research, we restrict the scope of robot knowledge to be: strings, which contain information about objects and activities. This includes physical properties of objects and verbal descriptions, which are usually assigned by a human to objects and strings that contain information about activities that includes verbal descriptions and activity components recorded in temporal sequence. Physical properties of objects are present in various forms; some of them are temporal, e.g., acoustic information, whereas others are static, e.g., curvature of a surface. Activities can also be represented in various formats. However, from a robot's point of view, they are not much different from each other, as they are all strings. As long as the robot knows how, i.e., has the program, to decode the string, they are considered the same. For the sake of simplicity, and also to avoid some of the less practically

useful philosophical debate, we consider that for a robot to know an object or an activity merely means to have information about the object or activity stored in its memory. Therefore, for a human, a verbal description of an object is quite different from the size, weight or other physical properties of the object as language usually roots deeply to the *meaning* or *concept* of the object. To say it in a different way, language provides a representation of the object, as one needs to understand the word in the description to extract information embedded in the description. On the other hand, for a robot, a verbal description is the same as sensor collected physical properties. They are all byte strings stored in the memory. If a keyboard can be viewed as a sensor, then key strokes are sensor inputs and character strings are the sensor outputs.

Sensory information is the first step for humans and robots to perceive the external world. However, there is much more than simple perception when humans live in the real world. We not only perceive things, but also understand them. We build concepts out of percepts. In the previous paragraph, we stated that for a robot, to know is to have information stored in its memory. This is clearly not the case for humans as someone could easily memorize a physics formula without knowing its meaning, or someone could read out a poem word by word without understanding it. Could we do something similar when we build robots? What does it mean for a robot to understand a concept? Could a robot really understand anything, not just react based on programs run on the robot?

We believe those are interesting questions. However, rather than addressing them in this research, we would like to take a functional point of view. We believe robots are built to help humans to perform certain tasks which are either impossible or inconvenient for humans to perform. Therefore, if it is beneficial to have a robot to know there are similarities between a fork and a spoon, i.e., the distance measure between a fork and a spoon is less than a fork, say, a chair, then we should program a distance evaluation function, which always returns a smaller number when a fork and a spoon are compared. In such a measure, we define that identical objects have a distance measure equal to zero, and this measure returns only nonnegative

numbers. If a robot can use this function to perform tasks better, then we may say that the robot knows, or is able to infer, that forks, spoons and probably dinner knives, belong to one group, whereas tables, chairs and bookshelves belong to another.

In a nutshell, *robot knowledge* is information about objects and activities, in string format, stored in a robot's memory. For a robot, to know an object means to have information about that object stored in its memory. Relations between objects help robots to perform tasks better. However, relations are developed or discovered based on information acquired through sensors.

2.2 Knowledge Extraction

In the previous section, we have presented our definition of robot knowledge. We have emphasized that grounding knowledge to sensor data is essential to this work. This section introduces how knowledge can be extracted from sensor data, and presents the data sample set that has been used for this work.

2.2.1 Data Type and Extraction

Sensors produce data in many formats. Typical sensors available to robots are sonar, laser range finder, weight scale, CCD camera, infrared, odometers, etc. In general, these sensors can produce results in two categories: direct measures and derived measures. For instance, a weight measure of an object is a direct measure of a weight scale; an RGB histogram of an image is a derived measure of a CCD camera. The accuracy of direct measures depends on the accuracy of the sensor. The accuracy of derived measures depends on the accuracy of the sensor and the algorithm used to produce the measure. Therefore, it is our hope to set standards on algorithms used to produce derived measures, and to regulate the format of these measurements. Therefore, comparable results can be obtained.

To test the suitability of the direct/derived data measure taxonomy and the possibility of the measurement standardization, and to provide a solid ground for our research, we collect sample data for objects and measure their properties. Since

the prototype of this framework is to be applied to kitchen robots, we collect our sample data from kitchenware, e.g., forks, spoons, knives, cups, plates, etc. CCD cameras are readily available and the produced images are information intensive. We choose to work with images as our starting point.

Since the 1970s, image analysis and retrieval has been an active research area in database and computer vision [5]. Image retrieval is primarily text-based in research of database, whereas in research of computer vision, it relies on visual properties of the data. In the early 1990s, content-based image retrieval (CBIR) was proposed. Ref. [5] summarized CBIR as:

> In CBIR, images are automatically indexed by summarizing their visual contents through automatically extracted quantities or features such as color, texture, or shape. ... Since the inception of CBIR, many techniques have been developed along this direction, and many retrieval systems, both research and commercial, have been built.

The basis of CBIR is feature extraction, as shown in Figure 2.1. Typical features are color, texture, shape, sketch, etc. Normally, each feature has more than one representation. For instance, color histogram and color moments both represent color features. There are several ways to compute a color histogram. Contrast, uniformity, coarseness, roughness, frequency, density, and directionality are examples of texture features. They contain information about the structural arrangement of surface elements and their relationship to the surrounding environment.

To ensure system flexibility, we build our framework to support as many feature extraction techniques as possible. Determining the most appropriate set of image features for knowledge sharing in the context of robot knowledge would be interesting as well. However, to make the research concrete, we predefine an image feature set and collect a data sample set.

2.2.2 Data Sample

Sixteen objects, including four bowls, one cup, two forks, three knives, two plates, and four spoons, are selected. Two images are taken of each object, one from the top view, and the other one from the side view. All images are taken against a

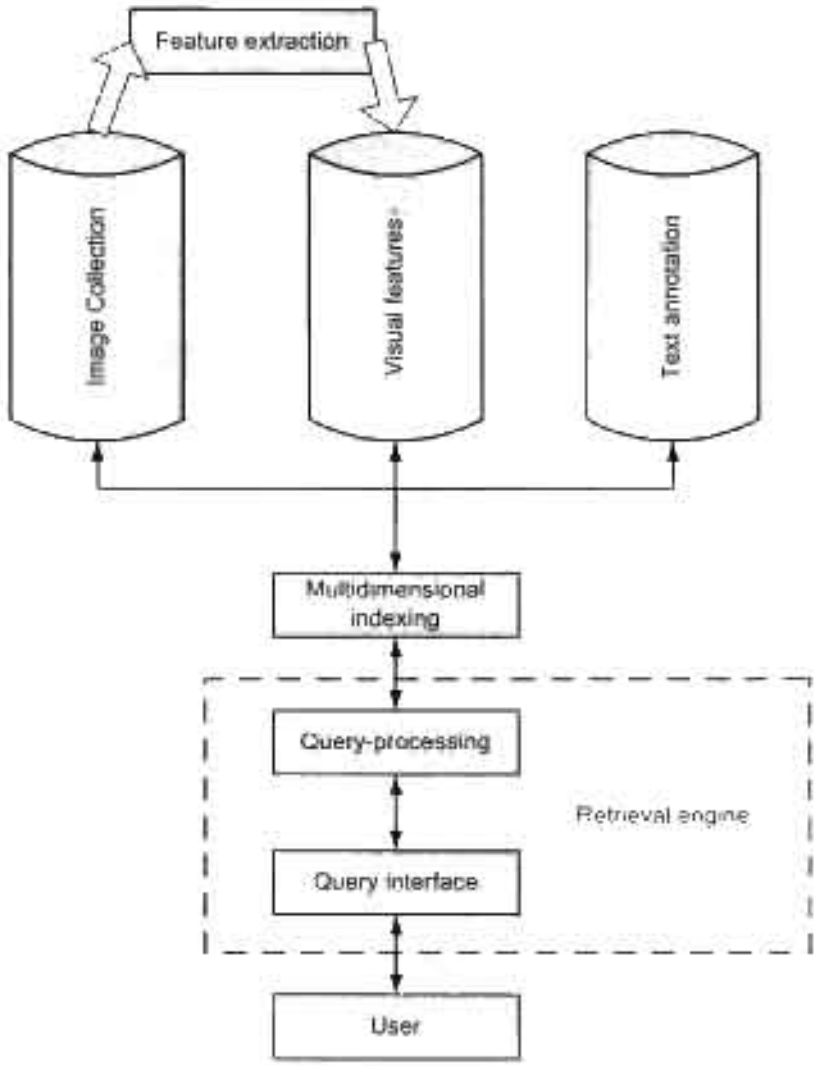

Figure 2.1 An Image Retrieval System Architecture

white background and objects are later manually segmented. Images are stored in JPEG format at the size of 616-by-480 pixels. For each image, the histograms of its color in RGB channels are computed, with 256 bins for each channel. The image is then converted to HSV color space, where the histogram of the hue channel is computed, with 256 bins. The Sobel edge detection algorithm is applied to compute the distribution of edge orientations, and a histogram with 256 bins is obtained. The color image is then converted to a binary image. The perimeter and the area of connected components in the binary image are computed. In order to capture

the uncertainty of the real world, we have studied how these measures may vary under different picture-taking conditions, e.g., various lighting conditions. We use natural light (indoor sunlight), two incandescent lights, and a flash light as our light sources in this test. Four objects - a cup, a plate, a spoon and a knife - are imaged. Twenty images under these four different light settings are taken of each object. The RGB histograms, the hue histogram, the edge direction histogram, the perimeter and the image area are computed. Figure 2.2 shows an image of a bowl from our sample set. Figure 2.3 shows three concatenated color histograms of this image from each color channel. Figure 2.4 shows the edge histogram computed using the Sobel algorithm.

2.3 Knowledge Formulation

In order to build the Robot Share search engine, which supports a large amount of data and fast retrievals, data indexing is needed. The indexing structure is discussed in the next chapter. Knowledge representation is presented here.

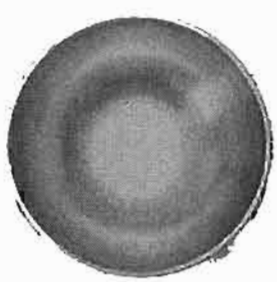

Figure 2.2. A Sample Image.

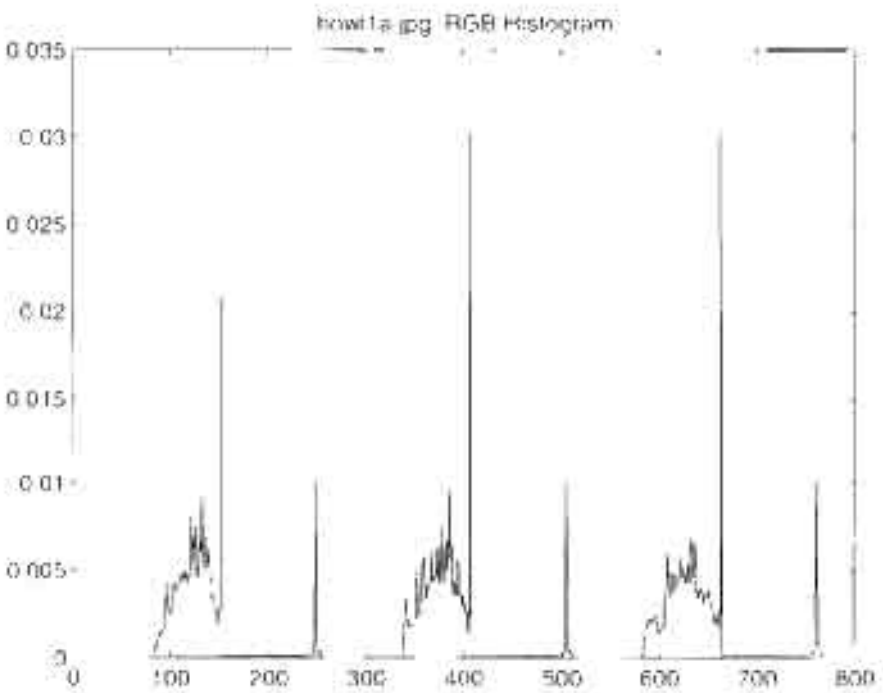

Figure 2.3 A Sample Color Histogram.

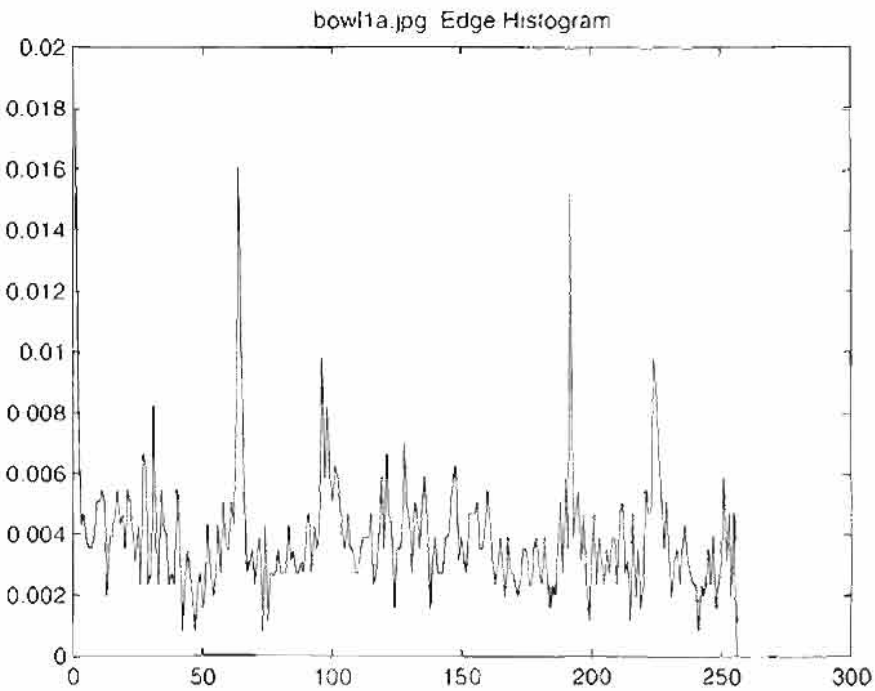

Figure 2.4. A Sample Edge Histogram.

Unlike a traditional database query system, where system architects also control the data source, Robot Share has the problem that it knows little about its data source. It is normal for a robot to know certain features about an object, and it is willing to share this knowledge with other robots though Robot Share. However, Robot Share may know little about this feature. Hence, Robot Share does not know how to process this knowledge. It is also possible that a robot wants to share knowledge of an object even though it does not know every feature of this object. In order to make Robot Share to support these two common cases, two knowledge transformations are required.

The first transformation takes place in robots, where knowledge is transformed from a robot's internal representations, which are probably only known to robots themselves, to a form such that they are understandable to other parties, e.g., Robot Share and other robots. The second transformation takes place in Robot Share, where knowledge, which is represented in the format produced by the first transformation, is then transformed into a representation that can be efficiently indexed.

2.3.1 The First Transformation

The purpose of the first transformation, from a robot's internal format to an open standard, is to transform knowledge in such a way that an unambiguous and widely-adoptable format is achieved. Two requirements need to be satisfied for this purpose. First, the data source of the transformation needs to be collected uniformly. We propose the idea of the asymmetric spatial-temporal coherence for objects. When a robot collects information about an object, i.e., measures its properties, we assume the robot does this in a uniform way such that all properties are measured with the least intervention among them. For instance, it is not desirable for a robot to measure one property of an object and somehow manipulate the object and then measure another property. Once all information is collected, the robot packages it tightly to maintain data integrity. Therefore, it is clear to

both the robot and Robot Share that information about one particular object is collected.

Having a clear distinction between an object instance and an object class is significant to this work for two reasons. First, Robot Share has deep roots in the concept of sensor data grounded knowledge, i.e., sensor data are the ground for all higher level knowledge structure. Therefore, knowing which instance a sensor data refers to is important to all higher levels, e.g., semantic level, knowledge structure formation. Second, it is desirable to support instance-based query in addition to the general class-based query. For example, to be able to detect that an image represents a human face is useful (the class-based query), but to be able to detect whose face it is (instance-based query) can be more useful for some applications.

We employ the standard Extensible Markup Language (XML) to represent knowledge as the result of the first transformation, as the XML format is widely used and accessible. We give a precise definition of the language our framework supports by using the Knowledge Definition Grammar (KDG). KDG is designed to be flexible enough to capture various type of knowledge and parser friendly. The definition of KDG is included in Appendix B. As all knowledge about objects in our framework is sensor grounded, even though KDG provides the ability to support virtually any type of object property, we define a set of XML tags to describe certain common object properties.

2.3.2 The Second Transformation

The purpose of the second transformation is to convert the easy-to-communicate XML format into a representation that is easy to index. Hence we can build the search engine efficiently. We take the vector space approach.

Every piece of knowledge in our system can be divided into three parts: text data, sensor data and meta data. Text data are provided by humans. They include the name, function, use and possible other related descriptions about an object. Sensor data are collected through sensors. They represent physical properties of an object. They are recorded by numerical values. For instance, the weight of an

object is usually recorded with a single numerical value, given a standard unit is used; the shape of an object can be recorded by a histogram of the direction of the object's edge, where a histogram is usually represented by a vector. Meta data are recorded when the object is measured by sensors. They contain information about collected sensor data. For instance, the location of where the object is encountered, the time when the object is encountered, the type/band/model of the sensor used to collect data, etc.

These three types of data can be indexed using two different approaches: different data types are either indexed separately, using multiple indexing structures, or they are combined and indexed by a uniform structure.

There are pros and cons to either of these two approaches. Separated indexing has the advantage that each type of data can take its own indexing method. The indexing method of a particular data type can be entirely based on the type of the data. For instance, LSI and reverse index are commonly used text indexing methods. They can be evaluated and selected for the text data indexing. Image data have a different set of properties. They can utilize some other indexing method. So do meta data. Finer grained discrimination is possible as well. Sonar and laser range finder generate output in different formats. Data generated from each of the two can hence be indexed differently. Data-class based indexing could result in a better retrieval performance, if queries contain only one type of data sample, i.e., text query, sonar query, etc. However, if a query contains more than one type of data, then the query has to be first divided into multiple smaller queries and the search engine issues each of these queries into each individual indexing structure accordingly. Retrieved results from each of the indexing structures are cross-processed before they are returned to the querying robot. One drawback of this approach is the high maintenance cost. Since multiple indexing structures needs to be implemented and data consistency between indexing structures needs to be maintained, the complexity of building and updating this system is considerably high.

If we combine all data types and use a single indexing structure, the multiple indexing structure data coherency problem is eliminated. Upon receiving a multiple fields query, we no longer need to worry about how to cross process results generated by separated subqueries. However, missing an indexing field in a query now becomes a problem. Many other problems exist in uniform indexing. For instance, we need to develop a technique that combines all three data types (text, sensor, and meta data) into one indexing structure. Even though some information carried by these data can be compromised during the grouping process, the end result must contain enough information to ensure a clear classification. Considering the verity of data types, the only sensible solution is to build the indexing structure in a high dimensional vector space where each object is mapped into a point in this space, i.e., every object is represented by a long vector. More consequences follow the choice of representation. This topic will be revisited in the next chapter, in which the two indexing mechanisms are further compared.

Regardless the choice of indexing structure selection, we need to develop techniques that map data into vectors. Mapping sensor data into vectors is straightforward enough as sensor data are naturally represented by either a single numerical value or a vector. We only need to arrange them in such a way that they can be indexed efficiently. Meta data can be processed in a similar manner, as most of information can be represented by numerical values. For instance, time can be unambiguously represented by a UNIX time string, locations can be represented by a GPS coordinate, and sensor type can be represented by an index of the sensor into a sensor database, such as 1 for Fairchild 9000, 2 for Serial No. 28753. Instead of using sensor models, logical sensor system frame can be employed.[1] Converting text into numerical vectors appears to be problematic on the first glance. However, techniques such as the LSI have been well studied. LSI converts documents into vectors in a mathematically meaningful way. Details of the LSI will be presented in the next chapter.

[1]Even though such a complete database does not exist today, we do not see any technology barrier that prevents researchers from building one, given enough demand.

The length of a data vector needs to be controlled. As will be explained in the next chapter, vector space index in high dimensions is much more problematic than in low dimensions. In another words, vector concatenated by a 256-*by*-1 color histogram, a 256-*by*-1 edge histogram and a few other data fields cannot be indexed efficiently without further process. Therefore, we have to shorten the length of some of these data fields, especially histograms. The next chapter will introduce various methods to achieve this goal of dimension reduction. For the purpose of this research, we have evaluated four different methods: Fourier coefficient representation, polynomial coefficient representation, statistics representation and moment representation. Using each of these methods, we reduce 256-element vectors to 5-element vectors. For the Fourier coefficient method, the first five coefficients are selected. For the polynomial coefficient method, a fourth-order polynomial approximation is used. For the statistics method, variance, median, standard deviation, median of the first half values, and median of the second half values are used. For the moment method, the first five moments are computed. Since all algorithms have the same reduction rate, i.e., they all transform a 256-element vector to a 5-element vector, we would like to select the transformation that mostly preserves the distance measures of data in its original space.

Distance measure complicates this comparison as distances are determined by the selected measure. It is possible that one transformation provides better results than another transformation under one distance measure but provides worse results under a different measure. It is also possible for a transformation function and distance measure pair to perform better on one set of histograms but perform worse on another one, e.g., the pair good for edges are bad for colors. Indexing structure selection further complicates the comparsion. Indexing structures can divide the search space based on a particular subset of a vector, instead of using all information stored in the vector. Hence the distance between two points, which are represented by two vectors, becomes less meaningful. For instance, suppose the L1 distance d is measured between two points in a 10−dimensional space. When the indexing structure is constructed, only the first two elements in a vector are

considered.[2] In this case, the distance obtained through the L1 distance measure d becomes less meaningful than, for instance, the sum of the two differences of the first and the second elements from each vector.

Transformation function and distance measure selection problems do not stop at the indexing structures. Since one of the goals of this research is to build a search engine for robots, we have to further consider how transformation function and distance measure may affect query processing, especially because queries can come in with missing fields. We may conclude it is sufficient to judge our design decision solely based on retrieval performance, assuming we can ignore any consideration on computational cost. However, knowing the exact consequence of adjusting parameters in internal states of Robot Share could help us to improve the entire system. Performing a complete study on all parameter selections may not be possible given the complexity of the system and the variety of data. Nevertheless, it is possible to gain some insights into those problems once enough experiments are conducted. This leads to interesting future research.

Transformation functions and distance measures have been examined through a series of experiments. Edge histograms are selected for examination. In order to simulate the real world uncertainty, we add noise into the data set. For each histogram, noise drawn from a Gaussian distribution with mean equals to 20% of the histogram mean is added. Thirty noise-added samples are created for each histogram; 480 histograms are obtained. We then apply four vector-length-reduction transformations to the sample set and obtain four sets of samples. Each sample set contains 480 5-element vectors. For each new sample set, we measure distances between every two vectors and produce a confusion matrix for each measure. Four distances measures are applied in this process, L1, L2, KL and weighted L1. Sixteen 480-*by*-480 confusion matrices are produced (please see Appendix A for printouts.) Confusion matrices give us a good visual to this experiment. Among these distance

[2]This is possible as when using a k-d-tree as the indexing structure. One algorithm selects the split based on the spread of data in that dimension, so when the first two elements in vectors have spreads larger than the rest of dimensions, they will be repeatedly selected.

measures, L1 and weighted L1 are better (in term of keeping separated classes separate) than K-L and L2 measure. Fourier Coefficients and polynomial coefficients are better vector length reduction techniques than the other two. To perform a qualitative analysis, we have created a perfect classification confusion matrix sample, in which we manually assert the value of each element in this matrix. Then we compare 16 confusion matrices against this sample by using the normalized correlation. We have concluded that polynomial coefficient representation is the most reasonable choice. Figure 2.5 shows an example of an how edge histogram is approximated by a polynomial.

2.3.3 Example

We conclude this section with a simple example from our data set to demonstrate the experiment. A robot notices an object in a kitchen. The robot measures physical

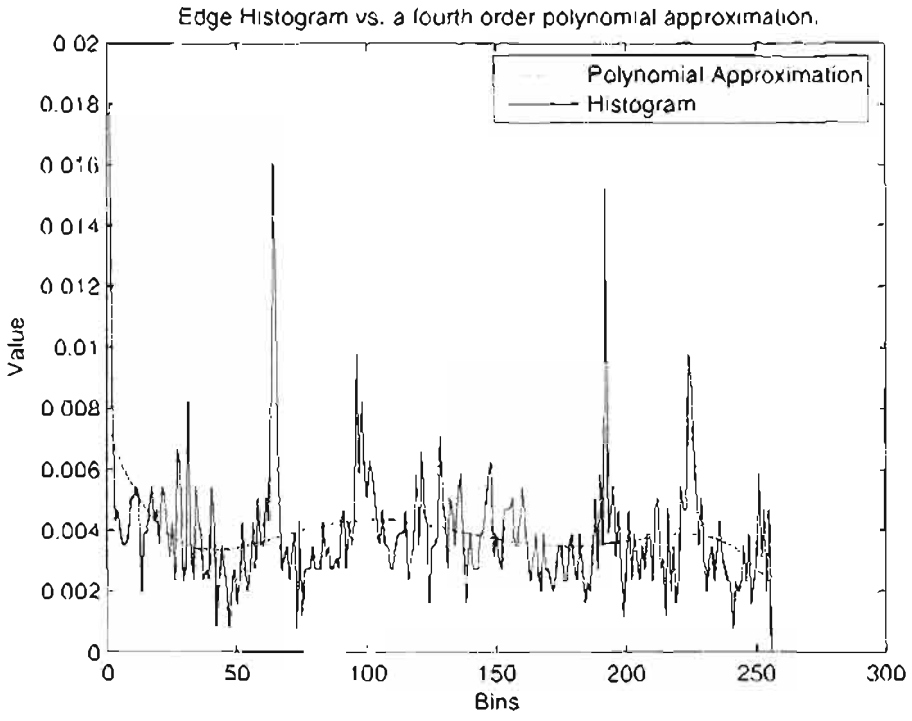

Figure 2.5. An Edge Histogram and its Polynomial Approximation.

properties of the object as the following: the length is 12.5 centimeters; the height is 5 centimeters; the width is 12.5 centimeters; and the weight is 2.5 ounces. Two images are taken of this object, one from the top view and one from the side view. Images are segmented; color and edge histograms are computed. The robot has learned from a human that this object is a "Knife with black handle." Since the robot wants to share information about this object, in addition to organizing and uploading these data into a web space, the robot registers this information at Robot Share. It then packages the object into a Robot Share understandable XML file and sends it to Robot Share.

After Robot Share receives this file, Robot Share parses the XML file and constructs a set of vectors to capture information stored in this file. The text part, i.e., "Knife with black handle," goes through the LSI process, and becomes a four-element vector. The two images go through a sequence of image processing procedures. Color histograms and edges are produced. Then dimension reduction techniques are applied, and histograms are reduced to short length vectors. Dimension and weight measures are extracted. The final result can be viewed in Table 2.1. Using this information, Robot Share builds an index for these data and stores them in the Robot Share database.

2.4 Activity Knowledge

The previous section has discussed how object knowledge can be represented. Object knowledge is one type of knowledge we would like to share. Another type of knowledge is activity knowledge. Activity knowledge represents a much broader range of knowledge. If we consider object knowledge is mostly about object identification and classification, then activity knowledge can be used not only for identification but also for the execution of activities.

One of the most challenging problems to solve in activity knowledge sharing comes from the uncertainty of the real world environment. For instance, it is possible that one robot specifies a procedure that performs a certain activity in its own environment, i.e., under a set of conditions. It is difficult for a second

Table 2.1 A Sample Object.

Field	Value
red1:	[1.8464e-011 -6.6396e-009 4.4481e-007 3.0729e-005 4.8032e-004]
green1:	[2.2378e-011 8.7258e-009 8.0668e-007 8.9936e-006 -2.2540e-004]
blue1:	[2.2060e-011 8.5507e-009 7.6566e-007 1.3780e-005 -4.5634e-004]
red2:	[1.4508e-012 2.5196e-009 -9.8969e-007 8.7476e-005 2.4578e-004]
green2:	[3.1170e-012 1.8227e-009 -9.2036e-007 8.8420e-005 3.8085e-004]
blue2:	[4.2376e-012 1.3480e-009 -8.7116e-007 8.9412e-005 5.5129e-004]
edge1:	[-1.2498e-010 6.3984e-008 -1.0713e-005 6.4745e-004 -0.0063]
edge2:	[-1.3743e-010 6.9901e-008 -1.1638e-005 6.9704e-004 -0.0070]
LSI:	[0.0509 0.2674 0.2571 0.4403]
phy_vec:	[6.0190 1.6906 12.3570 0.4998]
weight:	0.4244
filename:	'knife2a.jpg'
desc:	'Knife with black handle'

robot to blindly adopt this procedure to perform the same task as the second robot lives in a different environment. Two approaches can solve this problem. The first approach focuses on the environment. A detailed description of a robot environment can be recorded and shared with the activity description. When the learner robot learns the new procedure, the learning robot checks if it has a similar environment with the sharing robot. The second approach is to bring intelligence into the learning robot, i.e., when it receives an activity description, it adopts the knowledge selectively. For instance, if certain instruction in the activity description is not suitable for the learning robot's own environment, it generates and executes substitute instructions with equivalent functions. Either of these two approaches requires an extensive amount of research. Approach one assumes the ability of environment recognition and representation, and approach two assumes the ability of environment recognition, local planning and activity result prediction. All of these requirements point to interesting topics for future research.

The Robot Share research focuses on activity knowledge identification. The problem statement can be summarized as one robot records a sequence of human

body movements and queries Robot Share for information to identify the activity that being performed by these movements.

Collaborating with Prof. Dillmann's humanoid robot research group at the University of Karlsruhe, we have obtained data generated by the *VooDoo human motion capture system* [24, 25], which gathers data of the human configuration over time, resulting in 3D trajectories for every modeled limb and joint angle of the human body. In *VooDoo*, the human body is represented by 19 4-*by*-4 transformation matrices, where each matrix describes the state of a limb joint. In each transformation matrix, the upper left 3-*by*-3 submatrix describes the rotation of the joint, the right most column describes the movement of the joint. (See [25] for a complete discussion of the *VooDoo* system representation.) We exploit two of these matrices: one that describes the trunk of the body transformation and the other that describes the right forearm transformation, from each activity instance frame. The motion description is based on six values from each of the two transforms: three diagonal elements of the rotation matrix and three translation components. This results in 12 feature vectors. We then approximate the trajectory of every feature field across frames of an activity instance by a fourth order polynomial. The end result looks similar to object knowledge that has been described in the previous sections, i.e., each activity is represented in 12 vectors that can be indexed using techniques will be described in the next chapter.

CHAPTER 3

KNOWLEDGE SEARCH ENGINE

3.1 Robot Knowledge Search Engine

As described by Frieden and Kuntz [13], the three main tasks of a search engine are to (1) match query keywords with related material on the web, (2) rank web documents according to relevance, and (3) provide pointers to the documents. Arasu et al. [1] set their major emphasis to be the creation of scalable index structures. Note that search engines for human created web documents try to make the linkage among web pages explicit and exploit this to create structure indexes.

When adopting their strategy to the robot world, one of the major issues is the diversity of the data format. Unlike text documents, which are focused by current web search engine, sensor data produced by robots exists in many ways, e.g., images, sound wave files, etc. These data need to be processed before they are entered into database. Multiformat data also lead to various difficulties in database design. Research in multimedia database shares some common problems we have. Ortega-Binderberger et al. [5] pointed out that a multimedia database need to provide four functionalities:

- **Multimedia Object Representation.** Techniques or models to succinctly represent both structure and content of multimedia objects in databases.
- **Content Extraction.** Mechanisms to automatically or semiautomatically extract meaningful features that capture the content of multimedia objects and that can be indexed to support retrieval.
- **Multimedia Information Retrieval.** Techniques to match and retrieve multimedia objects on the basis of the similarity of their representation.

- **Multimedia Database Management.** Extensions to data management technologies of indexing and query processing to effectively support efficient content-based retrieval in database management systems.

The previous chapter has addressed the first two issues. This chapter focuses on the latter two.

3.2 Knowledge Harvesting

In this first generation robot search engine research, we do not foresee a major role for web crawlers. Even if web pages that contain robot information exist, the meta data are not available to determine what pages to download and what is of interest in them (e.g., there are no words to count and no lexicon to help define any semantics.) There is no popularity measure and no standard place to find things (e.g., specific sites, in homepage, etc.) We decide to let robots register with the Robot Share and provide direct meta data and links.

In the previous chapter, we presented the format for knowledge communication between a robot and Robot Share. Since the XML file presented in the previous chapter solely contains the data of objects, a few extra fields are helpful for Robot Share. The most important one is the link to the web address, where the original data can be found. Storing information about the robot that registered the information could be helpful as well. Therefore, three fields are added into a knowledge registration XML file, the identity of the robot, the time of this registration, and the link to the web page, where original data are stored.

3.3 Knowledge Query

Before we present the object retrieval architecture of Robot Share, we first discuss some related technologies. Section 3.3 is dedicated to search queries, section 3.4 discusses distance measures, and section 3.5 introduces indexing structures.

3.3.1 Query Type

Similarity retrieval can be divided into whole match and partial match. In the first class, every object is considered in whole, i.e., the query is considered as an object and then matched against objects stored in the database. In the second class, the query is compared to portions of objects in the database. Therefore a portion of certain object can be returned to a query as a response. Research in CBIR provides us good examples of both of these two classes. Projects dedicated to photographic image retrieval are mostly in the class of whole match [11, 14] and image analysis projects usually support partial/subimage match [28]. Partial match systems usually bring the problem of data segmentation, which itself contains a lot of variations. Robot Share supports whole match.

3.3.2 Query Interface

Query interfaces of retrieval systems vary. Query-by-example, query-by-feature, and other miscellaneous methods have been demonstrated [5]. In a query-by-example system, the query is treated as an object as every other object stored in its database. The query is first analyzed to extract its features. Extracted features are then used to query the database. Most CBIR systems belong to this group. In a query-by-feature system, the user specifies a set of feature and their values for the retrieval system to match. Keyword, image-annotation and meta data based image retrievals belong to this group. Since the 1990s, multimedia, especially image, query languages have been developed in the database research society. Most of them are SQL extensions or variations, including PSQL, Spatial SQL, QL/G, MOQL, etc. Ref. [27] contains an survey of them. Query-by-feature systems usually require support from a dedicated query language. For Robot Share, query-by-example is supported. No specific query language is employed in Robot Share.

3.3.3 Search Type

Unlike text based retrieval system, a multimedia retrieval system does not rely on exact queries. Instead, multimedia retrievals are similarity based. Using interfaces described above, the user specifies a feature set, the retrieval system then tries to find data that have a similar feature set. Three types of search exist.

- **Range Search.** Find all data in which feature f_1 is within range r_1, feature f_2 is within range r_2, etc. Query-by-feature systems work with this type of search.

- **k-Nearest-Neighbor Search.** Using distance measure D, find k objects that are closest to the query object. Note this type of query requires at least k objects to be returned, regardless of their actual distance to the query template. This could potentially result in returning objects that are very different from the query template. Returned objects are usually ranked based on their similarities to the query template.

- **Within-Distance (or α-cut).** Using distance measure D, find all objects that are within distance α to the query template. In contrast to the k-nearest-neighbor class, α-cut search could result in no return or returning the entire database. Returned objects are usually ranked based on object similarities.

The k-Nearest-Neighbor search and the α-cut search are both supported in Robot Share.

3.4 Similarity Functions

3.4.1 Distance Measures

The previous section has introduced a set of query types. Among them, k-nearest-neighbor and α-cut both require the distance measure between two objects to be computed. This section presents some commonly used methods to measure object distances.

A similarity function is a mapping between pairs of feature vectors and a positive real-valued number, which represents the similarity between two objects. Some of

the commonly used dissimilarity measures are listed (See Ref. [5, 30] for a more complete discussion.)

Let the descriptor be represented as an m-dimensional vector $\bar{f} = [f_1 ... f_m]^T$. Given two objects, I and J, let $D(I, J)$ be the distance between the two objects as measured using the descriptors f_I and f_J.

- **L1 distance**

$$D(I, J) = |f_i - f_j| = \sum_{k=1}^{m} |f_{k,I} - f_{k,J}|$$

- **Euclidean distance (L2 Distance)**

$$D(I, J) = \|f_I - f_J\| = (f_I - f_J)^T (f_I - f_J)$$

- **L_∞ distance**

$$D(I, J) = max|f_{k,I} - f_{k,J}|$$

- **Weighted L1 distance**

$$D(I, J) = \sum_{k=1}^{m} \frac{|f_{k,I} - f_{k,J}|}{\sigma_k}$$

σ_k is the standard deviation of the kth feature component in the database.

- **Mahalanobis distance**

$$D(I, J) = (f_I - f_J)^T \Sigma^{-1} (f_I - f_J)$$

Σ is the covariance matrix that
$\Sigma = E[(f - \mu_f)(f - \mu_f)^T]$ and $\mu_f = E[f]$.

- **Kullback-Leibler (K-L) divergence (relative entropy)**

If f is a normalized histogram, then

$$D(I, J) = \sum_{k=1}^{m} f_{k,I} log \frac{f_{k,I}}{f_{k,J}}.$$

L1, L2, and L_∞ distance belong to the category of Minkowski-form metric, which has the general form:

$$d_{q,t}^r = [\sum_{m=0}^{M-1} |h_q(m) - h_t(m)|^r]$$

where h_q and h_t are two objects with M features. (In the case of L_∞, it is interpreted as $\lim_{p\to\infty} D^p$.)

L1 distance is commonly known as the Manhattan distance, city block distance, or walk distance. When it is used to compare color histograms, it is also known as the Histogram Intersection [38].

L2 distance is refered as Euclidean distance. It can be written as

$$D(I, J) = \|f_I - f_J\| = \sum_{m=0}^{M-1} |f_J(m) - f_J(m)]^2.$$

One character of a Minkowski metric is that it compares the proportion of a specific feature within object q to the proportion of the same feature within object t, but not to the proportions of other similar features. For instance, when a Minkowski metric is applied to compare color histograms, [5]:

> the distance between a dark red image and a lighter red image is measured to be the same as the distance between the same dark red image and a perceptually more different blue image.

The main computational cost in this family is due to computing the power functions.

The weighted L1 distance is a special case of the weighted Minkowski family. It has the general form as:

$$D^{(p)}(I, J) = [\sum_{i=1}^{d} \omega_i |f_I[i] - f_J[i]|^p]^{\frac{1}{p}}.$$

Weighted Minkowski matric contains a weighting parameter ω for every individual feature. The standard deviation of the kth feature, σ_k, is a common selection for this weighting parameter.

Mahalanobis distance is a special case of the quadratic-form metric, which has its general form as

$$D(I, J) = (f_I - f_J)^T A^{-1}(f_I - f_J).$$

Its use on the color histogram can be found in the IBM QBIC system [14]. For some application, it generates more desirable results than matrices from the Minkowski family as the Minkowski family compares only like bins, whereas quadratic-from metrics consider the cross-relation of the bins.

Kullback-Leibler (K-L) divergence is defined only for probability distributions. Unlike others, relative entropy is not technically a distance measure as it is not symmetric, and it does not satisfy a triangle inequality. (Mahalanobis distance satisfies these two requirements when A is positive definite.) For query purposes, the first argument is set to the query template where the second argument comes from the database.

Ref.[5] discusses various properties of these distance measures in depth. A comparison of some of these matrices applied in image retrieval is presented. Despite manually selected and relative small sample sizes used in their study (which contains no more than 50 images in each of the tests), they provide some informative results, including simple metrics such as L1 and L2 alone with the more sophisticated Mahalanobis distance all give reasonable performances. K-L measure is not included in this test, but reported with good performance elsewhere in ref. [5].

3.4.2 Experiment Data

In the previous chapter, we have discussed the combined effort of distance measure and vector-length-reduction transformations. We have covered a series of experimental data in the form of confusion matrix measures. We then conclude that weighted L1 distance and polynomial coefficient provide the best performance. Here we compare the four original histogram samples, without reducing the vector length, to the same perfect sample and collect measures. This test is needed for search result ranking.

3.5 Feature Indexing

Indexing schemes are commonly used in multimedia database queries. The operation required to perform content-based search in such systems are computationally expensive. It is also known that indexing in multimedia databases is very different from indexing in text-based databases as multimedia data are stored in form of feature vectors. The previous chapter introduced the knowledge representation, array of short vectors, Robot Share uses. This section explains reasons behind this selection by explaining the difficulty of building indexing structures over long vectors. We then review a few commonly used indexing structures and discuss pros and cons of each of them.

3.5.1 Curse of Dimensionality

The most problematic issue caused by long feature vectors is the curse of dimensionality. This effect has been noticed by researchers from various domains. This phenomenon appears as numerous geometric properties that hold in low-dimensional spaces no longer hold in high-dimensional spaces. As ref. [5] explains:

> ...in two dimensional a circle is well-approximated by the minimum bounding square; the ratio of the areas is $4/\pi$. However, in 100 dimensions the ratio of the volumes becomes approximately $4.2 * 10^{39}$: most of the volume of a 100-dimensional hypercube is outside the largest inscribed sphere - hypercubes are poor approximations of hypersphers and a majority of indexing structures partition the space into hypercubes or hyperrectangles.

For example, the widely used R-tree indexing schemes become inefficient for α-cut queries using the L2 distance. As ref. [5] explains:

> [R-tree indexing are inefficient] as they execute the search by transforming it into the range query defined by the minimum bounding rectangles of the desired search regin, which is a sphere centered on the template point, and by checking whether the retrieved results satisfy the query. In high dimensions, the R-trees retrieve mostly irrelevant points that lie within the hyperrectangle but outside the hypersphere.

Another problem of high-dimensional space is that points randomly sampled from the same distribution appear uniformly far from each other and each point sees itself as an outlier. For instance, we can have an example such as the following:

- We first generate 20,000 independent 100-dimensional vectors, with the features of each vector independently distributed as the standard Normal random variables.

- Then we compute the Euclidean distance from a random vector from the same distribution to all vectors in the database.

- We observe that the minimum distance between the query point and database sample is above 10, the average distance is about 14, and the maximum average distance is above 17. Figure 3.1 shows the shape of the distribution.

Comparing Figure 3.2 to Figure 3.3, with vector lengths of one and five, respectively, we can see the distance distribution in high-dimensional spaces differs from distributions in low-dimensional spaces. This effect makes α-cut queries very sensitive to the choice of the threshold in high-dimensional spaces. For instance, when the threshold is smaller than 10, no result is returned; with a threshold of 12.5, the query returns 5.3% of the database; and when the threshold is increased to 13, 14% percent of the database is returned.

3.5.2 Feature Selection

The curse of dimensionality can also be seen in the field of patten classification [27]. We can view an object query as classifying a new object into a known category. In this view, the classification error should decrease when additional measurements are applied. However, this is not always true in practice. When a classifier is constructed, there may not be enough sample to train the classifier, i.e., determine the most appropriate parameters for each feature, e.g., the threshold for a α-cut query of certain feature. Specifically, the classifier would be well tuned for the training set, but would fail when new instances are presented. Therefore, we would need to minimize the feature set for a classifier to minimize the number of unknown parameters.

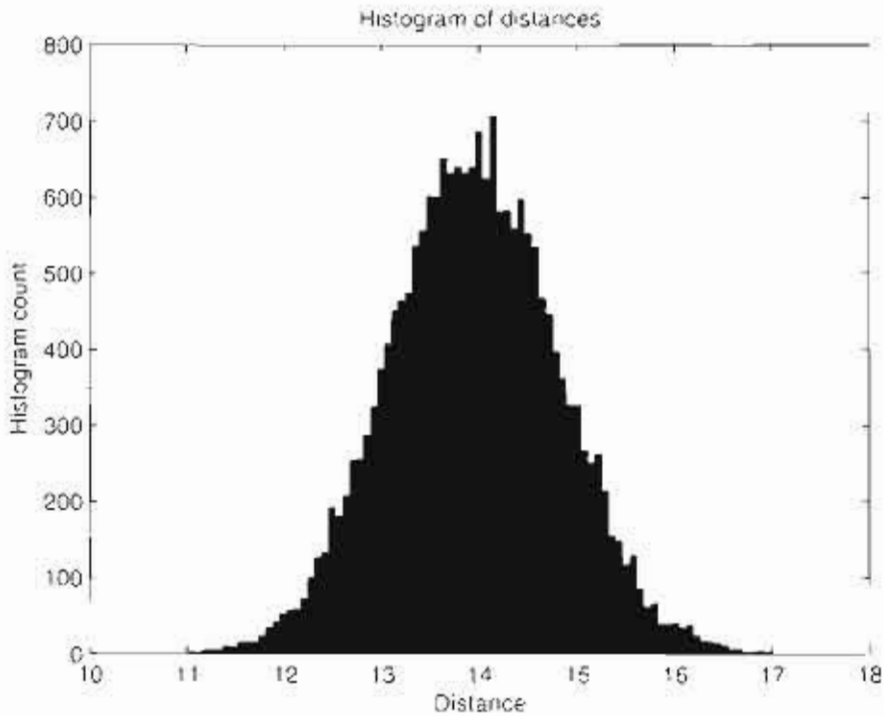

Figure 3.1. Distances Between a Query Point and Database Samples Sample Size = 20,000; Vector Dimension = 100; Bin Number = 100 $D_{min} = 10.22$, $D_{max} = 17.41$, $D_{avg} = 13.94$

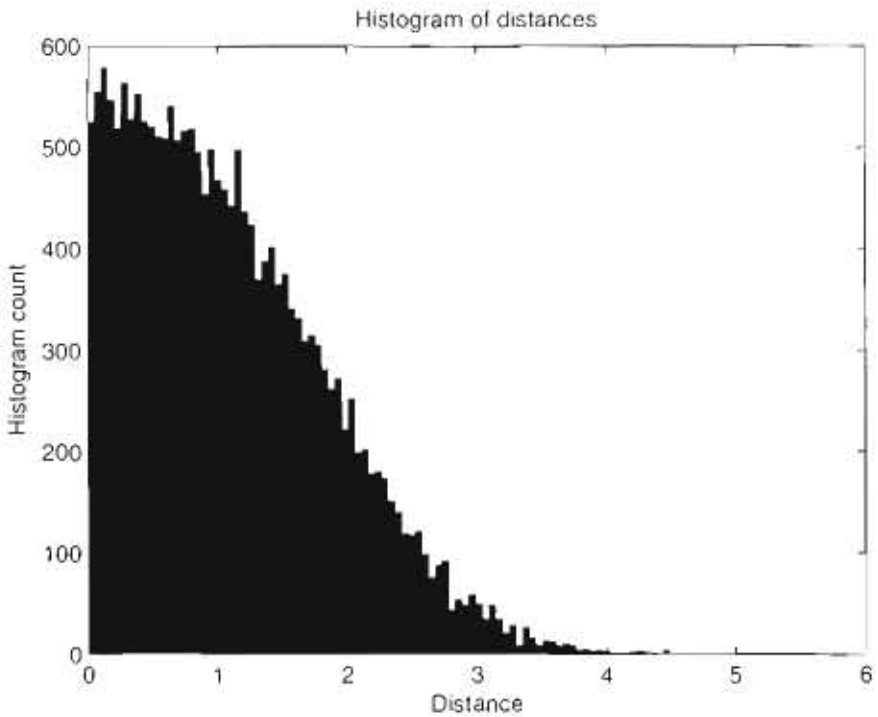

Figure 3.2. Distances Between a Query Point and Database Samples. Sample Size = 20,000; Vector Dimension = 1; Bin Number = 100. $D_{min} = 0, D_{max} = 5.16, D_{avg} = 1.11$

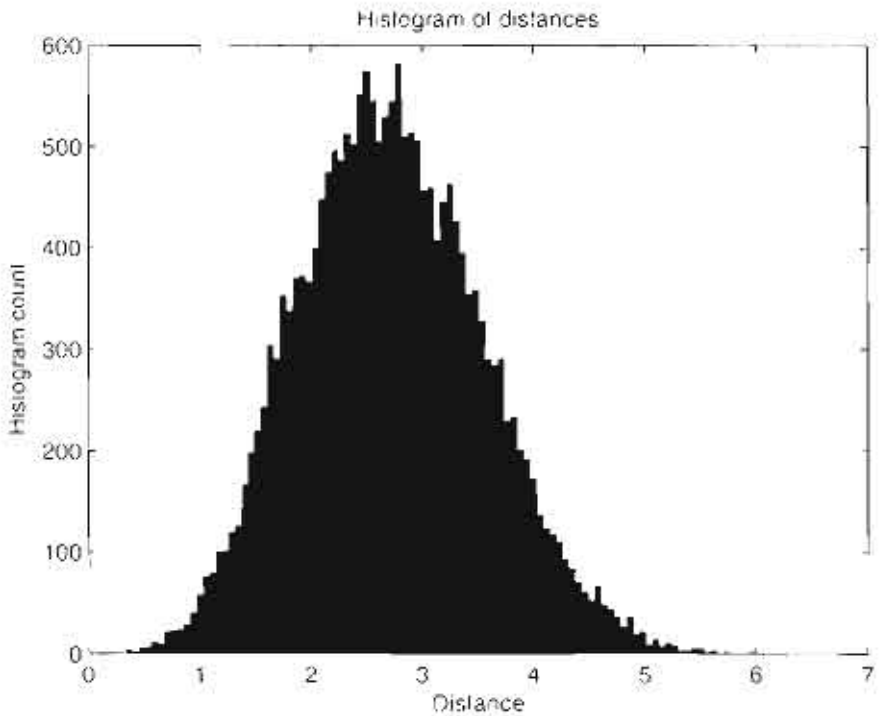

Figure 3.3. Distances Between a Query Point and Database Samples. Sample Size = 20,000; Vector Dimension = 5; Bin Number = 100. $D_{min} = 0.28, D_{max} = 6.05, D_{avg} = 2.72$

Feature selection can be viewed as a typical searching problem formed as: Select $d \leq D$ from

$$V = \{v_i | j = 1, 2, ..., D\}$$

arriving at

$$U = \{u_i | i = 1, 2, ..., d\}$$

where V is the complete feature set, each u_i is an element of V, and U maximizes a criterion function, which is the probability of correct classification. Clearly, brute force feature selection is not feasible, as the permutation of M choose N grows exponentially with the growth of M and N, e.g., selecting 10 features out of 100 would necessitate evaluation of more than 10^{13} feature sets. Thus, a computationally feasible method must be used.

Feature selection hence has been studied and a few algorithms have been proposed. Among them, the sequential forward selection (SFS), the sequential backward selection (SBS) and their derived variations are popular methods. Ref. [27] states the following:

> SFS is a bottom-up, hill climbing search procedure, where one feature class is added at a time to the current feature set. At each iteration, the feature class to be included in the feature set is selected from among the remaining feature classes such that the new feature set yields the greatest possible value of the criterion function. ... The two major drawbacks ... are the local peak problem and the ridge problem.

The local peak problem and the ridge problem are common to all hill climbing search algorithms. The first one occurs when there are multiple maxima and the search is trapped in one local maximum. The second one occurs when the path to a local maximum is not within available search directions. A typical hill climbing algorithm stops when a local maximum is achieved without searching for the global maximum. It also stops when none of its operation yields a higher score in its evaluation function, i.e., reaches a ridge. In case of SFS, it reaches a local maximum such as adding no other feature to a particular feature set V_1 could results a higher evaluation index. It also fails when a high evaluation index is reachable only when multiple features are added at once.

SBS is the top-down counterpart to the SFS. The complete feature set is the starting point. At each stage, one feature is removed until removing no feature can result a higher evaluation index. It suffers from same problems as SFS.

To overcome, or reduce the chance of getting into, the ridge problem, generalized sequential forward selection (GSFS) and generalized sequential backward selection (GSBS) are developed. In these methods, instead of adding/deleting one feature at a time, multiple features are added/deleted. In general, GSFS and GSBS produce better results than SFS and SBS, but, again, with higher computational costs.

Another variation is the plus L take away R selection (LRS), where SFS and SBS are applied interchangeably. Again, better results are reported, but with an increased computational cost.

Nonfixed L and R sizes at each step can be utilized, i.e., instead of adding or removing a fixed number of features at each step, a floating number of features are considered. This further increases the possibility of finding the global maximum.

More discussion on techniques such as stochastic methods and neural networks on feature selection can be found in ref. [23, 27]. Even though progress has been made in this area, selecting a small yet representative set of features for general applications is an open problem.

3.5.3 Dimensionality Reduction

Feature selection is a useful technique that helps to overcome the curse of dimensionality. However, as some of the most popular indexing methods, including R-Tree and Quad-Tree, perform suboptimally when the dimensionality of the feature space barely exceeds 10, we usually need to further decrease the dimensionality of the feature space. Three classes of methods have been investigated.

Variable-subset selection works just like the feature selection. Certain features are discarded upon indexing. A small set of useful ones are left to keep the dimensionality low. The main problem of this approach is the error induced by approximating the original vectors with their lower-dimensionality projections. Techniques in this group are Karhunen-Loeve transform (KLT) [15], singular value

decomposition (SVD) [22] and principal component analysis (PCA) [27]. A variable-subset selection step discards dimensions that have smaller variance. It can be shown that when Euclidean distance is used to measure distance, the distance between original vectors is closer than the distance measured with their projections using above techniques. The main disadvantage in above techniques is the computational cost, as KLT, SVD and PCA are data-dependent. They are poorly suited for dynamic databases, in which items are added and removed on a regular basis.

Multidimensional scaling is another technique used in this field. In general, this technique tries to remap the original feature space R^n into R^m $(m < n)$ using m transformations, each of which is a linear combination of appropriate radial basis functions. Various implementations exist in this category. The drawback of these algorithms is the high computational cost. Hence they are not suitable for dynamic databases.

Geometric hashing [4] consists of hashing from a high-dimensional space to a very low-dimensional space (usually one or two dimensions.) As hashing functions are not data-dependent, the metric properties of the hashed space can be significantly different from the original one. The design of a good hashing function becomes increasingly difficult as the dimensionality of the original space grows.

To summarize, a good dimension reduction algorithm should maintain the distance measure when the feature set is transformed from a high-dimensional space to a low-dimensional space. It also must be computationally efficient if dynamic databases need to be supported.

3.5.4 Indexing Structures

Ref [5] contains an intensive discussion on the topic of indexing structure. A handful indexing scheme are reviewed. In general, we distinguish indexing structures in three ways: (1) vector space indexes versus metric space indexes; (2) recursive partitioning methods versus projection-based methods; and (3) spatial access methods (SAM) versus point access methods (PAM).

Vector space indexes represent objects and feature vectors as sets of points in a d-dimensional vector space. Metric space indexes pairwise distances between objects in a set instead of indexing objects. It tries to capture the metric structure of the search space. Recursive partitioning methods organize the search space into a tree like structure. Projection-based methods employ algorithms that perform searches on the projections of database point onto a set of directions. SAM indexes spatial objects, e.g., lines, polygons, surfaces, solids, etc. PAM index points in multidimensional spaces. More detailed discussion of indexing structures can be found in ref. [5, 16, 35].

Among these techniques, recursive partitioning methods have been widely used and reported with good results. The three most commonly used categories in this group are quad-trees, k-d trees, and R-trees. Castelli [5] explained this as the following:

> Quad-trees divide a d-dimensional space into 2^d regions by simultaneously splitting all axes into two parts. Each nonterminal node has therefore 2^d children, and, as in the other two classes of methods, corresponds to hyperrectangles aligned with the coordinate axes. K-d trees divide the space using $(d-1)$-dimensional hyperplanes perpendicular to a specific coordinate axis. Each nonterminal node has therefore at least two children. The coordinate axis can be selected using a round-robin criterion or as a function of the properties of the data indexed by the node. ... R-trees divide the space into a collection of possibly overlapping hyperrectangles. Each internal node corresponds to a hyperrectangular region of the search space, which generally contains the hyperrectangular regions of the children. The indexed data is stored at the leaf nodes of the tree.

Quad-trees are a large class of indexing structures. Besides the classic algorithm described above, its variation includes region quad-tree, point quad-tree, etc. However, quad-trees are not well-suited for high-dimensional ($d > 10$) indexing [5]. For instance, when $d = 20$, the quad-tree becomes very sparse, i.e., most of its nodes are empty. In higher dimension, hyperspheres are not well-approximated by hyperrectangles. Therefore quad-trees are not suitable for α-cut and nearest-neighbor queries.

The k-dimensional tree, known as k-d tree, is another commonly used hierarchical indexing structure. As its name suggests, it gives better performance than quad-trees in high-dimensional space. Constructing a k-d tree is more costly than constructing a quad-tree in general. Furthermore, when the data node cannot be efficiently split, k-d trees suffer from the utilization imbalance problem. A modified version, k-d-b tree, was proposed [31]. Unlike the original version, it supports dynamic node insertion and deletion. K-d trees are expected to give reasonable performance for $d \leq 20$.

R-trees and their large variations are probably the most-studied multidimensional indexing structures. An R-tree splits the space using hyperrectangles rather than hyperplanes. The properties of R-trees differ from the previous two families as it allows overlapping rectangles. Derivations of R-trees include R^+-trees, R^*trees, packed R-trees, X-trees, VAMSplit R-tree, S-tree, etc. It has been reported that R-tree shows enough efficiency in up to 20 dimensions [11].

3.5.5 Indexing Text Data

Text indexing has a much longer history than multimedia data indexing. In fact, most early research in image retrieval was based on text-based retrieval though image annotation [5]. However, the researcher then realized image annotation based retrieval was limited, as to have a human annotate the ever increasing large amount of image data is not practical. Then we saw the birth and growth of CBIR. On the other hand, up to date, text-based retrieval has been more successful than image-based systems. Robot Share utilizes text indexing as well as content based image indexing.

We unify textual and multimedia sensor data in our system as we believe an integrated system is easier to maintain and gives a better performance. To achieve this goal, all text information needs to be represented in numerical vectors and a low dimensional representation is desired. As suggested in ref. [36], we investigated the latent semantic indexing (LSI) [8]. Ref. [36] explains LSI as:

> LSI works by statistically associating related words to the semantic context of the given document. The idea is to project words in similar documents to an implicit underlying semantic structure.

LSI tries to solve the synonymy and polysemy problems. The synonymy problem is that the same object can be described in multiple words. The polysemy problem is that a word can mean multiple things. There is a many-to-many relation that exists between objects and words.

The singular-value decomposition (SVD) is the main workhorse in LSI. The algorithm used in ref. [36] works in the following way:

- Construct the term $\times$ document matrix A, where the element a_{ij} represents the frequency of term i in document j. Therefore, each column of the matrix A is a term histogram of a document.

- Decompose A using the SVD:

$$A = U\Sigma V^T$$

  where $U^TU = V^TV = I, \Sigma = diag(\sigma_1, ..., \sigma_n), \Sigma$ is the singular value matrix of A.

- Then select k most significant dimensions from the original space. This is achieved by selecting the first k rows in Σ where σ_i has a higher value.

- Then the k-dimensional vector representation of the qth document is:

$$\tilde{X} = q^T U_k \Sigma_k^{-1}$$

  where q is the word frequency histogram of the qth document, U_k is a matrix composed of the first k columns in U, and Σ_k is a matrix composed of the first k rows and k columns of Σ.

Ref [8] does not provide a systematic approach in selecting k. The hope is k should be large enough so most "latent semantic" in one document can be captured. It is also needs to be small enough so data noise can be removed.

3.6 Robot Share Architecture

After reviewing the concept and related technologies of search engine, we are ready to present the Robot Share architecture. We first revisit the single indexing structure vs. multiple indexing structure discussion; then we give detailed discussion on each component in Robot Share.

3.6.1 Indexing Structure Consideration

In the previous chapter, we explained that representing objects in a vector space is a sensible approach for the construction of indexing structures. However, there was a debate on whether a unified single indexing structure (Figure 3.4.) is better than a set of small, potentially heterogeneous, indexing structures (Figure 3.5.) Considering the need of knowledge representation, a unified single indexing structure requires a long vector concatenated by a set of small vectors, where each small vector records certain property of the object. A set of smaller indexing structures requires every small indexing structure to take a small vector. Therefore there is no need to concatenate vectors, which represent different properties, into a long one. Hence objects are effectively represented by arrays of vectors. The previous chapter summarized the advantage of the unified long vector approach as simple, easy to implement and maintain and the advantages of the set of small vectors as more elegant and having better performance with incomplete queries. We see more evidence points that the set-of-small-vectors approach is a more feasible solution if more things are taken into consideration.

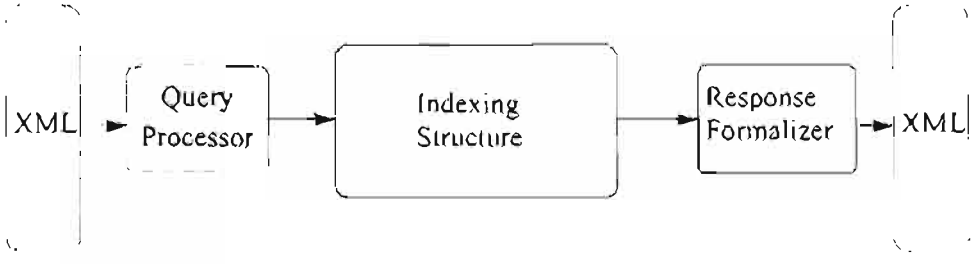

Figure 3.4. The Single-indexing-structure Architecture.

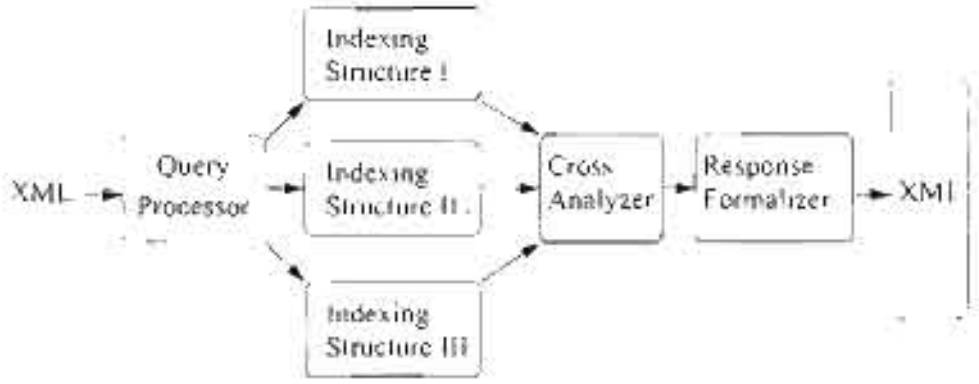

Figure 3.5. The Multi-indexing structure Architecture

First, the curse of dimensionality determines that the length of vectors used for indexing needs to be short. Even though various mathematical techniques help to reduce the size of long vectors, such as histograms, there is a limit on the amount of size reduction can be applied without losing a significant amount of information. One good way to measure the amount information that has been preserved after dimension reduction is to compare the confusion matrix that is generated using the original data, i.e., long length vectors, against the confusion matrix generated using reduced vectors. Since a confusion matrix is a good measure of the classifying power of a certain property of an object, comparing two confusion matrices is informative. If too much information has been lost, then this resulted representation loses the power of classifying object. For example, edge histograms are usually good at distinguishing objects, where each histogram can contain more than a hundred bins, hence each of these histograms is represented by a vector with more than a hundred elements (256 in our sample data). If we compare distances between histograms stored in our database, we can construct a confusion matrix, which shows how those histograms can be classified. A similar confusion matrix can be constructed using reduced vectors, e.g., approximate a histogram by a fourth-order polynomial and the five coefficients can be used to represent the original histogram. The second matrix can not be identical to the first, as some amount of information must be lost during dimensional reduction. However, if the second matrix is close enough to

the first matrix, then we conclude that the classifying power of the original data is mostly preserved [1]. A series of experiments have shown that transforming a 256-bin histogram to a 5-element vector using polynomial coefficient representation yields a reasonable performance. More aggressive reductions sacrifice from performance losses. However, in our sample data, each object contains eight histograms. Even if all of them are reduced to 5-element vectors, the length of the concatenated final vector approaches 50. Building a indexing structure supports a 50-dimensional space without suffering from the curse of dimensionality is a challenge.

Second, from a software engineering point of view, a single big indexing structure suffers from poor scalability. One of the most significant characteristics distinguishes Robot Share from other work in multimedia database or CBIR is the flexibility Robot Share aims at. Robot Share is designed to support as many data format/object representations as possible. However, if we commit to a specifically designed indexing structure for all data Robot Share takes, then it is very likely that the desired system flexibility will be lost. On the contrary, if we divide indexing structures into multiple ones, where each of them handles one type of data, it will be easy to add more object properties by adding more indexing components.

We had briefly mentioned before was separating indexing component into smaller ones could help us to improve the performance of each of these components. For example, in the current implementation, LSI is used to handle text data. Even though we report good retrieval results with LSI, it inherently has the problem of being too static, as adding any new document into the database requires a recompute on the entire structure. Any modification made to keywords requires a

[1] Note that we are not stating that a transformation from a high dimension to a lower one always decreases the classifying power. It is possible that in certain cases, a low-dimensional representation has more classifying power than its high dimensional counterpart. For two reasons. (1) data stored in the high dimension vector are interrelated, so information redundancy exists, (2) some portion of the high dimension data is noise. In other cases, the high dimension to low dimension transformation works more like a low pass filter, where useful information is extracted from the original data. However, for the purpose of this discussion (e.g., transforming a 256-bin edge histogram to a 5-element vector), we always see losing classification power rather than gaining it. Hence the difference between the two confusion matrices can always be interpreted as a loss of classifying power.

recompute as well. It has been noticed that a SVD decomposition, which is at the core of LSI, is computationally expensive. Therefore, we may need to reconsider if there is any substitute, as we can replace the text indexing component with the new module without worrying about the rest of Robot Share. Another example is finding better transformations for histograms. Different histograms come in different shapes, hence they are better if approximated using different techniques, e.g., polynomial approximation and trigonometry function approximation. If we separate the indexing structure into components, we can efficiently study each component hence achieving a better overall performance.

Robot Share supports both instance-based query and class-based query. Instance-based query focuses on finding additional information for a particular object. Class-based query focuses on object classification. Even though both of the single-big-indexing-structure and the set-of-small-indexing-structures are able to perform the instance-based query, the single indexing structure struggles to support class-based queries. For instance, there are four objects stored in the database, two bowls and two knives. Two bowls are represented by: $[1, 2, 2]$, $[10, 2, 2]$; and two knives are represented by $[1, 8, 8]$ and $[10, 8, 8]$. The confusion matrix measured using L1 is:

$$\begin{bmatrix} 0 & 9 & 12 & 21 \\ 9 & 0 & 21 & 12 \\ 12 & 21 & 0 & 9 \\ 21 & 12 & 9 & 0 \end{bmatrix}$$

Based on measured distances using L1, two bowls are grouped together in one group while two knives are also grouped together in another group. Figure 3.6 shows a k-d tree placement of these four vectors. In this case, suppose a bowl represented by $[1, 1, 1]$ is received for a query, even though the closest bowl, represented as $[1, 2, 2]$, is returned correctly; the other bowl, $[10, 2, 2]$, will be missed, and the knife, represented as $[1, 8, 8]$, is also incorrectly returned. The upshot is, indexing structure such as a k-d tree is best for the nearest neighbor query. Its performance on k-nearest-neighbor queries is largely data depended. Therefore, if we are able to separate a large k-d tree into a few smaller ones, we have more control over the search procedure, hence better retrieval performance can be achieved. For

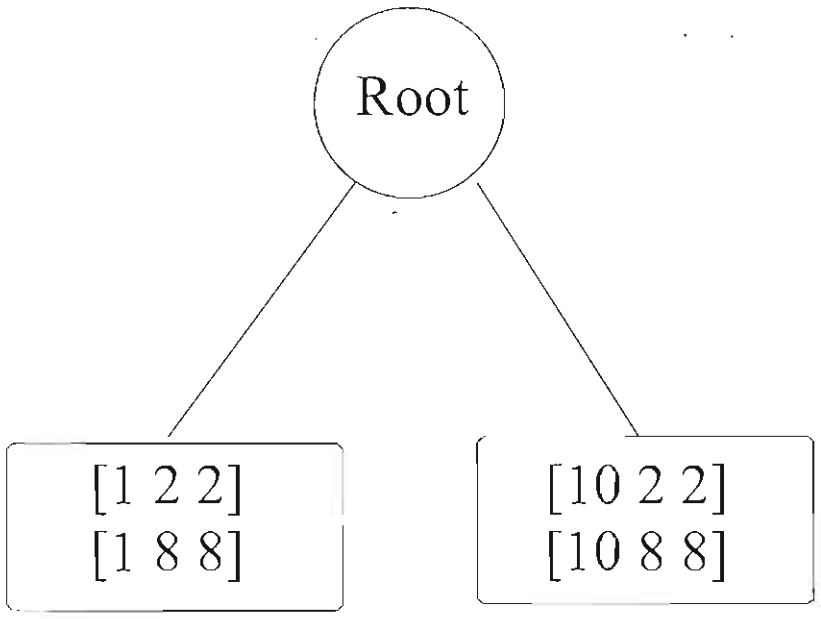

Figure 3.6. K-d Tree Structure of Two Bowls and Two Knives.

instance, we can use weight coefficients to put emphasis on properties that might better classify a certain group of objects than other properties. It can be viewed that we can use weight coefficient to dynamically enlarge or shrink the space in different dimensions. For instance, if we want to ignore certain features in a query, we put a small weight coefficient for this feature. This is equivalent to shrinking the space in dimensions that represent this feature. In this case, coordinates of objects in these dimensions no longer matter, as distances between points are dominated by distances measured in other dimensions.

Based on these four reasons, even though the set-of-small-indexing-structure approach has disadvantages in its complexity, we build Robot Share using this approach.

3.6.2 Robot Share Component

In the current implementation, Robot Share is composed of four groups of components: a query processor, indexing structures, a cross analyzer and a response formalizer. In the future, a feedback analyzer can be added into the system.

The query processor is the first component in a Robot Share query process. It takes queries, in the format of XML files, and translates them into an array of vectors and sends each vector to a corresponding indexing structure to find matches. The query processor functions as a simple XML parser as it converts data stored in XML to vectors, and computes various derived features from raw data stored in the XML file. For instance, for object knowledge, it computes color and edge histograms and transforms them into low dimensional representations. It also computes the vector representation of text information into vectors using LSI. In the future, the query processor could be built more intelligently so it not only parses data but also preprocesses them. For example, currently, robots perform image segmentation if a query contains an image. We may later add an image segmentation component into the query processor.

Another interesting topic to be studied is to have the query processor to discover underlying relations between information stored in different fields in the same object. For example, when an object comes with an image and a text description as a "yellow bowl," a color histogram is computed from the image. We know there is an underlying relation between the word "yellow" and the shape of the color histogram. Developing a systematic approach to discover all relations cross feature fields is an open problem. If a such an approach is developed, among other things, it can help Robot Share to approximate missing fields in both queries and data entries stored in its database, and possibly improve the retrieval performance.

The second group of components is indexing structures. They are arguably the most significant group of components in Robot Share. They take inputs from the query processor in form of vectors, and produce ordered lists of objects. They sort objects using measures between the query sample and objects stored in Robot Share and return the sorted list.

In the current implementation, 11 indexing components are created for object knowledge processing. Six of them are built for color histograms (two images of an object, three color channels in each image); two of them are built for edge histograms (two images for each object); one of them is built for text data produced

from an LSI process; one of them is built for dimensional information of the object, i.e., length, height, width, and the cube root of the product of the three; the last indexing structure is built for the weight measure of an object. These properties are summarized in Table 3.1. K-d trees have been used to index all of these fields except the weight measure, which uses a binary tree. In all k-d trees, branch dimensions are selected in the round-robin fashion. Branching starts from the left most element of a vector. This decision is made due to the fact that the eight histograms in one object are all approximated by polynomials, in which high order terms contribute more to the shape of the polynomial. Text data are processed by LSI, which has the same property that high order terms capture more information than low order ones. All indexing components return 15 items for each query, except the LSI text indexing and dimensional properties component, in which 30 items are returned.

For activity knowledge, in contrast to the 11 index structures developed for object knowledge sharing, 12 index structures are constructed for activity recognition as 12 trajectories are selected from each activity. K-d trees are used to index all trajectories. For the reason described above, branch dimensions in each k-d tree are selected in a round-robin fashion. Each indexing component returns five items

Table 3.1. The Robot Share Component Length Summary.

Component	Vector Length	Returned items
Red Channel (im1)	5	15
Green Channel (im1)	5	15
Blue Channel (im1)	5	15
Red Channel (im2)	5	15
Green Channel (im2)	5	15
Blue Channel (im2)	5	15
Edge Orientation (im1)	5	15
Edge Orientation (im2)	5	15
Text Data	4	30
Dimensional Properties	4	30
Weight	1	15

for each query. Items are then sent to the next module in Robot Share, the cross analyzer.

The cross analyzer takes item lists from each indexing component, "cross analyzes" them and produces a single sorted list. Cross analyzer creates the sorted list based on a weighted summation of all query-item distances. The cross analyzer first creates a list containing all received items. It then computes distance from the query sample to every item in the list. The distance measure is a weighted L1, which can be expressed as, the overall distance $D(A, B)$ between two objects A and B is equal to:

$$D(A, B) = \sum_{i=1}^{k} |w_i d_i(A_i, B_i)|.$$

In this equation, k equals to the number of fields presented in the query, w_i is the weight coefficient of the ith component, and d_i is the distance between ith components in the two objects. All $d_i(A_i, B_i)$ are computed using the L1 distance measure, where

$$d_i(A_i, B_i) = \sum_{j=1}^{k} |A_{i,j} - B_{i,j}|.$$

For all image histograms represented by polynomial coefficients, k equals to 5; for the LSI indexed text field, k equals to 4; and for the singleton weight field, k equals to 1. We are at the very beginning stage of developing a systematic approach of computing weight coefficients w_i. The ideal ranking order is query dependent. It is related to the intentional use of the knowledge, the content of the database, and the content of the query template. To find optimal weight coefficients, information about query robots must be taken into consideration. Currently, a static analysis approach is taken. We design experiments for various data conditions and query types. In each experiment, we evaluate Robot Share performance using the standard information retrieval measures: precision and recall [41]. We then search for weight coefficients that maximize these measures. The searching algorithm is an n-dimensional binary search, which is a good compromise between simplicity and performance.

There are two special cases making the distance computation more complicated. They are both related to missing fields in objects. The first case is: the query comes incomplete, e.g., a query that has no text description or misses one image. In this case, only indexing structures that associate with presented fields are used. As briefly mentioned in the previous section, if Robot Share is able to discover the underlying relations between fields, it is possible for Robot Share to fill some of these missing fields. We also consider objects stored in Robot Share that contain missing fields. In this case, ignoring information contained in queries by not using indexing structures associated with these fields would certainly yield a poor performance as (a) there are objects stored in Robot Share containing these fields and some of them could be what the querying robot looks for. Not using information stored in these fields can negatively impact the ranking of these objects. (b) Robot Share should utilize as much information in a query as possible. However, if Robot Share uses all indexing structures, then the distance between fields in the query sample and objects stored in Robot Share has to be estimated for objects with missing fields. This distance is approximated by measuring the query sample to a pseudo object, which approximates a missing field with the sample mean of that field. This approximation may not be a very good choice as when the sample variance is large, the sample mean becomes less representative of the actual value of an object. Future research is required to solve this problem better.

Once distances between the query sample and all items returned by indexing structure are computed, the cross analyzer sorts the item list based on these distances and sends the sorted list to the next module: the response formalizer.

The response formalizer takes input, which is a sorted object list from the cross analyzer, and generates an XML file that is understandable to the querying robot. The size of returned files, i.e., the length of the returned list, should be large enough so there is a high chance for the querying robot to find the information it needs in the returned file. The file also needs to be reasonablely small so (a) the file transmission can be done in a small amount of time and (b) after receiving the file, a robot can determine if any useful information can be found in this file quickly. To

find a compromise between these two requirements, there are two questions to be answered: (a) how many results should be returned, and (b) what information does one result contain? In the current implementation, for object knowledge, every indexing components returns either 30 or 15 items to the cross analyzer, and there are 11 indexing components in Robot Share, so the number of items generated by the cross analyzer ranges from 30 to 180. If all indexing components return the same set of items, then the length of the list cross analyzer generated is 30. If all indexing components return lists that are mutually exclusive to each other, then the length of the the list generated by the cross analyzer is 180. It is simple for Robot Share to adjust the length of its returned list. As if more return items are desired, Robot Share can increase the items returned by each indexing component; if fewer items are demanded, Robot Share can cut off the return list created by the cross analyzer. As for the content of a result, the most important information is URL links to web addresses, where original object data can be found. It also would be useful that certain object information, which is stored in Robot Share, could be presented to a querying robot along with the URL. So the querying robot can quickly scan through the list to eliminate objects that are not interesting to the robot. The format of returned XML files can be expressed in a simple grammar. The definition of the grammar is included in Appendix C.

CHAPTER 4

KNOWLEDGE EXPLOITATION

In the previous chapter, we have discussed the architecture of Robot Share and reasons behind a number of design choices. This chapter is dedicated to performance studies of Robot Share. One can realize that giving a complete evaluation to a complex system like Robot Share is a challenging task. To make this task manageable, we limit the scope of our study and divide the evaluation into small areas. Each area focuses on one performance aspect of Robot Share. These performance tests give us a comprehensive overview of the system and help us to study future improvement.

4.1 Object Knowledge Experiment

This section presents Robot Share performance on object knowledge. In all of these experiments, we have used common information retrieval performance measures, precision and recall, to evaluate our system. Precision in information retrieval is defined as the following:

$$precision = \frac{|relevantdocument \cap retrieveddocuments|}{|retrieveddocuments|}$$

It is a measure of the percentage of results that are desired in the total retrieved list. Recall in information retrieval is defined as

$$recall = \frac{|relevantdocument \cap retrieveddocuments|}{relevantdocuments}$$

Recall is the percentage of desired retrieval results in the entire database. Neither precession nor recall alone can indicate the performance of an information retrieval system[1]. But by combining the two, a comprehensive evaluation can be reached.

4.1.1 General Performance

We first examine how Robot Share performs in general. Two hundred samples are randomly selected from the Robot Share database. They are used as query templates to query Robot Share. The Robot Share sample database contains 480 sample objects that are derived from 16 real world collected sample objects. Each object duplicates 29 copies, and 20% random noise is added to each copy. All samples in the Robot Share database, including query samples, are complete, i.e., there is no missing information in either queries or database samples. Weight coefficients used for distance measure, in the cross analyzer, are set to one. All items returned from Robot Share are retrieved. Relevant items are siblings of images that are duplicated from the same sample object with added noise. One result, measured in precision and recall, is presented in Table 4.1.

We can observe that the recall for this experiment reaches its highest possible value, one. Due to the relatively large number of retrieved documents, and the relatively small number of relevant items in the database, the precision is on the low side.

We then define relevant items as items from the same class, i.e., bowls, knives, etc. In this setting, the range of relevant items are enlarged. This test examines how Robot Share perform on class-based queries.

From Table 4.2 we can see that since the average number of relevant items grows and the number of retrieved stays unchanged, the precision grows. For the same reason, the recall drops.

---

[1]Precision approaches one (the highest score it can reach) when the number of retrieved document approaches zero; recall reaches one (the highest score it can reach) when the number of retrieved document approaches the size of the database. Neither of the two situations satisfies a user's need though the measure is high.

Table 4.1. Performance Test 1.1a

	Min	Max	Median	Mean	Variance
Precision	0.2113	0.5882	0.3704	0.3677	0.0048
Recall	1	1	1	1	0
Retrieved	51	142	81	84.8400	321.5723

Table 4.2. Performance Test 1.1b

	Min	Max	Median	Mean	Variance
Precision	0.3488	0.9178	0.6066	0.6117	0.0156
Recall	0.2500	1	0.5583	0.5668	0.0240
Retrieved	55	130	80.5000	83.2200	239.8097

4.1.2 Query with Missing Fields

Robot Share is designed to be robust enough to handle incomplete queries, i.e., queries that miss certain fields. Robot Share also handles incomplete records, i.e. missing certain fields in objects from the Robot Share database. In the case of an incomplete query (Table 4.3 shows an example of a query representation that contains only one image and no weight measure), Robot Share utilizes information from fields that contain data. In the case of an incomplete record, Robot Share estimates values in a missing field using sample means from the database.

We first present performance data of Robot Share on incomplete queries. Results of queries that miss one image and weight measure are shown in Table 4.4. The first half of the table shows results from siblings relevant searches, and the second half of the table shows results from class searches. (All subsequent tables follow this format.)

Results can be interpreted as the following. We measure precision using all objects returned by Robot Share, and the number of items returned depends on the number of indexing components used for a query. In the case of missing fields in a query, the number of involved indexing components reduces. Therefore the overall number of returned items reduces. So the precision of searches grows. Compared

Table 4.3. A Query With Only One Image and No Weight Measure.

Field	Value
red1:	[1.8464e-011 -6.6396e-009 4.4481e-007 3.0729e-005 -4.8032e-004]
green1:	[2.2378e-011 -8.7258e-009 8.0668e-007 8.9936e-006 -2.2540e-004]
blue1:	[2.2060e-011 -8.5507e-009 7.6566e-007 1.3780e-005 -4.5634e-004]
red2:	
green2:	
blue2:	
edge1:	[-1.2498e-010 6.3984e-008 -1.0713e-005 6.4745e-004 -0.0063]
edge2:	
LSI:	[-0.0509 0.2674 0.2571 0.4403]
phy_vec:	[6.0190 1.6906 11.3570 0.4998]
weight:	
filename:	'knife2a.jpg'
desc:	'Knife with black handle'

Table 4.4. Performance Test 2.1

	Min	Max	Median	Mean	Variance
Precision	0.2439	0.6818	0.4478	0.4550	0.0078
Recall	1	1	1	1	0
Retrieved	44	123	67	68.6700	222.0415
Precision	0.3780	1	0.6701	0.6815	0.0193
Recall	0.2500	1	0.5250	0.5250	0.0291
Retrieved	47	106	67	68.4800	177.4036

with Performance Test 1.1, the average precision grows from 0.3677 to 0.4550 for sibling searches and 0.6177 to 0.6815 for class searches. For the same reason, the measured recall drops for class searches.

Then we test queries with no text description or dimensional measures. Results are in Table 4.5. For the same reason of further reducing the number of returned items, search precision grows for sibling searches. Even though the number of indexing components stays unchanged from the previous test, both text indexing and

Table 4.5. Performance Test 2.2

	Min	Max	Median	Mean	Variance
Precision	0.3544	0.8824	0.5000	0.5195	0.0117
Recall	0.9000	1	1	0.9840	5.0460e-004
Retrieved	34	79	59	58.9400	112.5216
Precision	0.3718	0.9048	0.5864	0.6089	0.0183
Recall	0.2500	1	0.3778	0.4393	0.0317
Retrieved	39	90	63	61.9800	85.9996

dimensional indexing return more items than other components. Hence removing the two reduces the number of returned items.

Then we test queries with text description. Results are in Table 4.6. From the confusion matrix presented in the previous chapter, we know text description is a good instrument for classifying objects. Measured precisions and recalls have proved this again. Using only text description, the retrieved length is short, and both precisions and recalls are high.

Then we test queries containing only dimensional data and weight measures. Results are in Table 4.7. This test shows the worst result in this group. Both precisions and recalls are low in either sibling searches and class searches. However, these results are not due to a high retrieved number as in previous tests. We hence conclude dimensional and weight measures are less efficient for object classification when the two are used in isolation.

Table 4.6. Performance Test 2.3

	Min	Max	Median	Mean	Variance
Precision	0.5000	1	1	0.9000	0.0404
Recall	1	1	1	1	0
Retrieved	30	60	30	36	145.4545
Precision	1	1	1	1	0
Recall	0.2500	1	0.3333	0.4342	0.0477
Retrieved	30	60	30	35.7000	139.9091

Table 4.7 Performance Test 2.4

	Min	Max	Median	Mean	Variance
Precision	0.0732	0.5526	0.2558	0.2786	0.0150
Recall	0.1000	0.7000	0.3667	0.3863	0.0245
Retrieved	36	44	42.5000	42.0900	4.0221
Precision	0.0682	0.9524	0.6551	0.6258	0.0473
Recall	0.0500	0.5833	0.2583	0.2886	0.0159
Retrieved	36	44	42	41.7700	3.9567

To summarize these results, two conclusions can be made: (1) since the number of returned items effects the measured search precision and the number of indexing components involved in a query determines the total number of returned items, removing certain indexing structures can increase the measured precision (2) Some features are more useful for classifying objects than others. In the above tests, image features and text features show higher classifying power than dimensional and weight data.

We then test Robot Share performance with incomplete records. First we randomly select 160 objects from the database (that is one third of the entire database) and remove all information obtained from the first image, i.e., color histograms and edge orientation histograms with mean values from the data sample then evaluate the retrieval performance. Results are presented in Table 4.8. In this test and all subsequent tests on incomplete records, queries are always complete.

Table 4.8 Performance Test 2.5

	Min	Max	Median	Mean	Variance
Precision	0.2055	0.5000	0.3409	0.3382	0.0033
Recall	1	1	1	1	0
Retrieved	60	146	88	91.4900	289.4039
Precision	0.2973	0.8462	0.5584	0.5560	0.0138
Recall	0.2667	1	0.5667	0.5581	0.0156
Retrieved	69	150	90	92.9500	261.4217

These results show no surprise. Compare with results from Performance Test 2.1, which contains complete data, average precisions in both sibling search and class search drop about 9%.

Then we reset the Robot Share database to its original state, i.e., all records are complete, and randomly select 160 objects, replace text data and dimensional data with sample means of each of these fields, respectively. The performance is presented in Table 4.9. The average precision for sibling searches further drops from the previous test. However, the average precision for class searches grows. Since the average precision is still lower than results from Test 2.1, we think this still shows losing information in database degrades search performance.

We again reset the Robot Share database to its original state and randomly select 160 objects, replace all of their data with sample means except one set of color histograms, one edge orientation histogram and text data. Results are presented in Table 4.10.

More information is removed from the database. Precisions for both sibling searches and class searches drop. However, since text and image data contain most information needed for classifying objects, the precision drop is not dramatic.

From these three tests, we can see that losing information in database items generally degrades search performance, especially measured precisions. However, since certain features bear more classifying power than others, losing information in less significant features degrades performance to a smaller extent.

Table 4.9. Performance Test 2.6

	Min	Max	Median	Mean	Variance
Precision	0.2479	0.4688	0.3125	0.3209	0.0018
Recall	0.8667	1	1	0.9933	3.3670e-004
Retrieved	64	121	95	94.4100	141.6383
Precision	0.2992	1	0.6392	0.5896	0.0283
Recall	0.3889	1	0.6111	0.6428	0.0233
Retrieved	65	127	90	92.5400	211.4630

Table 4.10. Performance Test 2.7

	Min	Max	Median	Mean	Variance
Precision	0.1961	0.5085	0.3409	0.3400	0.0034
Recall	1	1	1	1	0
Retrieved	59	153	88	91.0400	287.5337
Precision	0.3093	0.9474	0.5408	0.5623	0.0180
Recall	0.2500	1	0.5889	0.5965	0.0277
Retrieved	68	147	87.5000	91.2300	292.1587

The last group of tests in this section is on data with both incomplete queries and incomplete records. First, we estimate the first image information in one third of the database with sample mean, and test performance with queries with missing text information. Results are in Table 4.11. We see comparable results to Performance Test 2.5, where the query is complete but the database misses the same amount of information as in this test. Incomplete query reduces the number of returned items. Precisions stay the same.

We then test the performance with one third of the database missing one image and text information and queries miss the other image and weight measure. Results are in Table 4.12. The result is comparable to the previous one. Since more information is missing in this test, precisions should be lower than in the previous

Table 4.11. Performance Test 2.8

	Min	Max	Median	Mean	Variance
Precision	0.2389	0.5172	0.3333	0.3450	0.0036
Recall	0.8333	1	0.9667	0.9697	0.0014
Retrieved	58	113	87.5000	86.3000	140.6970
Precision	0.2545	0.8391	0.5143	0.5279	0.0181
Recall	0.2500	1	0.5167	0.5300	0.0359
Retrieved	58	114	89.5000	87.3700	145.4274

Table 4.12. Performance Test 2.9

	Min	Max	Median	Mean	Variance
Precision	0.2235	0.4906	0.3353	0.3351	0.0032
Recall	0.6000	0.9333	0.8000	0.8097	0.0058
Retrieved	50	94	76	73.8100	90.7211
Precision	0.2935	0.9565	0.5850	0.5843	0.0288
Recall	0.2250	0.9000	0.4750	0.4787	0.0219
Retrieved	46	102	75	74.3900	105.5938

one. However, the number of returned items reduces as well. Hence precisions stay at the same level as before.

In this section, we have presented Robot Share performance on incomplete data. In general we conclude that losing information in Robot Share database degrades search performances. However, as discussed previously, reducing the number of involved indexing components decreases the overall returned items, hence positively impacting measured precisions.

4.1.3 Ranking

All of our previous tests are based on one assumption that all results returned by Robot Share are used by querying robots. Therefore we constantly see high marks on recalls, but relatively low marks on precisions. For certain application, precisions are more important than recalls, and robots that send queries to Robot Share may be more interested in getting fewer results, which match the query template well, than getting a long and exhaustive list of data. N-nearest-neighbor searches are more suitable to their needs.

Knowing this need, we design another set of tests to evaluate the Robot Share ranking mechanism. Instead of analysis of all results returned for a query, we focus on the first 20 items. We measure precisions and recalls over them. To make this analysis more comprehensive, we have reused previous test settings. Test results are presented in Table 4.13, 4.14, 4.15.

Table 4.13 Performance Test 3.1 Part 1

	Min	Max	Median	Mean	Variance
	Complete Query and Complete Records				
Precision	0.4500	1	0.9000	0.8580	0.0185
Recall	0.3000	0.6667	0.6000	0.5720	0.0082
Retrieved	20	20	20	20	0
Precision	0.5500	1	0.9500	0.9200	0.0102
Recall	0.1083	0.6667	0.1667	0.2044	0.0173
Retrieved	20	20	20	20	0
	Incomplete Query Missing one image and weight measures				
Precision	0.0500	1	0.6500	0.6665	0.0400
Recall	0.0333	0.6667	0.4333	0.4443	0.0178
Retrieved	20	20	20	20	0
Precision	0.2500	1	1	0.9195	0.0171
Recall	0.0417	0.6667	0.1917	0.2675	0.0291
Retrieved	20	20	20	20	0
	Incomplete Query Missing texts and dimensional measures				
Precision	0.4500	1	1	0.9785	0.0073
Recall	0.3000	0.6667	0.6667	0.6523	0.0033
Retrieved	20	20	20	20	0
Precision	0.7500	1	1	0.9925	0.0014
Recall	0.1667	0.6667	0.2222	0.2562	0.0167
Retrieved	20	20	20	20	0
	Incomplete Query Missing all information but texts				
Precision	0	1	1	0.8900	0.0989
Recall	0	0.6667	0.6667	0.5933	0.0440
Retrieved	20	20	20	20	0
Precision	1	1	1	1	0
Recall	0.1667	0.6667	0.1667	0.1667	0.0143
Retrieved	20	20	20	20	0

Table 4.14 Performance Test 3.1 Part II

	Min	Max	Median	Mean	Variance
	Incomplete Query				
	Missing all but dimensional and weight measures				
Precision	0.0500	0.7500	0.2500	0.2705	0.0259
Recall	0.0333	0.5000	0.1667	0.1803	0.0115
Retrieved	20	20	20	20	0
Precision	0.0500	0.9000	0.4500	0.4375	0.0592
Recall	0.0083	0.2333	0.1000	0.0975	0.0029
Retrieved	20	20	20	20	0
	Incomplete Record				
	Missing one image				
Precision	0.1000	1	0.7000	0.7080	0.0498
Recall	0.0667	0.6667	0.4667	0.4720	0.0221
Retrieved	20	20	20	20	0
Precision	0.2000	1	0.9250	0.8645	0.0248
Recall	0.0667	0.6667	0.1889	0.2269	0.0178
Retrieved	20	20	20	20	0
	Incomplete Record				
	Missing texts and dimensional measures				
Precision	0.3000	1	0.6500	0.6765	0.0368
Recall	0.2000	0.6667	0.4333	0.4510	0.0164
Retrieved	20	20	20	20	0
Precision	0.4000	1	0.9500	0.9060	0.0179
Recall	0.2667	0.6667	0.2111	0.2478	0.0176
Retrieved	20	20	20	20	0
	Incomplete Record				
	Missing all but one image and texts				
Precision	0.2000	1	0.8500	0.6735	0.0629
Recall	0.1333	0.6667	0.4333	0.4490	0.0190
Retrieved	20	20	20	20	0
Precision	0.2500	1	0.8500	0.8555	0.0194
Recall	0.0667	0.6667	0.1778	0.2298	0.0204
Retrieved	20	20	20	20	0

Table 4.15 Performance Test 3.1 Part III

	Min	Max	Median	Mean	Variance
	Incomplete Query & Record Record missing one image Query missing texts				
Precision	0.3000	1	0.7500	0.7190	0.0279
Recall	0.2000	0.6667	0.5000	0.4793	0.0124
Retrieved	20	20	20	20	0
Precision	0.2000	1	0.9500	0.8810	0.0195
Recall	0.0667	0.6667	0.1667	0.2136	0.0134
Retrieved	20	20	20	20	0
	Incomplete Query & Record Record missing one image and text Query missing one image and weight measures				
Precision	0.1000	1	0.5500	0.5875	0.0385
Recall	0.0667	0.6667	0.3667	0.3917	0.0171
Retrieved	20	20	20	20	0
Precision	0.2000	1	0.9500	0.8805	0.0250
Recall	0.0333	0.6667	0.1722	0.2276	0.0181
Retrieved	20	20	20	20	0

Several observations can be made. First, we see a large uniform increase in search precisions, from 50% or below to 80% or above. This is expected as we now limit the size of the the retrieved list. It also reveals that Robot Share performs very well in terms of search result ranking. The only surprise from this part comes from searches with incomplete queries, i.e., queries with texts and dimensional data missing. The average precision reaches 98% in this test, which is even higher than tests that are conducted with complete queries. This implies weight measures not only have less classifying power than other fields, such as text data or images, but negatively impact object classification in our sample data. This conclusion seems unrealistic on first glance. However, if we look at the confusion matrix generated by weight measures, we see it mixes different objects badly. Searches on incomplete records show better performance as well. Precision nearly doubles when we compare results from Performance Test 2.5, 2.6, and 2.7, which are tests on the same conditions with full returns. This is expected, for the same reason that reducing the number of objects returned results in higher precision measures. Searches on incomplete records with incomplete queries also exhibit twice the performance, compared with searches returning the complete list on the same conditions. The measured precisions are still lower than searches on complete data, as expected.

As explained previously, we know that certain fields have more classifying power than others. Section 3.6.1 explained the benefit of separating indexing structure into small components so a different weight coefficient for each component can be applied. We have employed the static weight coefficients searching algorithm to find coefficients that maximize precisions. Using these coefficients, we see an increase on measured precisions. Results are presented in Table 4.16. Only siblings relevant searches are presented as class searches already show high marks on precision. Even though there are 11-indexing components currently implemented in the system, considering the computation cost of solving a 11-dimensional optimization problem, we decide to simplify this optimization problem by reducing it to a 5-dimensional one. We group color histograms in all channels into one, and edge histograms into another.

Table 4.16. Performance Test 3 2

	Min	Max	Median	Mean	Variance
	Complete Query and Complete Records				
Coefficient	[0 1 0 0 0]				
Precision	1	1	1	1	0
Recall	0.6667	0.6667	0.6667	0.6667	6.1007e-031
	Incomplete Query Missing one image and weight measures				
Coefficient	[0 1 0 0 0]				
Precision	1	1	1	1	0
Recall	0.6667	0.6667	0.6667	0.6667	6.1007e-31
	Incomplete Query Missing all but dimensional and weight measures				
Coefficient	[0 0 0 0 1]				
Precision	0.0500	0.7500	0.2500	0.3215	0.0259
Recall	0.0333	0.5000	0.1667	0.2143	0.0176
	Incomplete Record Missing texts and dimensional measures				
Coefficient	[0 1 0 0 0]				
Precision	1	1	1	1	0
Recall	0.6667	0.6667	0.6667	0.6667	6.1007e-31
	Incomplete Query & Record Record missing one image and text Query missing one image and weight measures				
Coefficient	[1 1 0 0 0]				
Precision	0.9000	1	1	0.9970	2.4343e-04
Recall	0.6000	0.6667	0.6667	0.6647	1.0819e-04

All six color histograms from two images take a same weight coefficient and two edge histograms from two images take the same weight coefficient. Coefficients are stored in a vector, in the order of: color histograms, edge histograms, text data, dimensional data, and weight measures. For instance, coefficient set [0 1 0 0 0] can be read as putting zero weight on color histograms, text data, dimensional data, and weight measures and only using edge histograms to classify objects.

Results from Performance Test 3.2 are interesting. Four out of five tests show the maximum average precision has been reached. The last one, where queries contain only dimensional and weight measures, the average precision is somewhat better than the one presented in Test 3.1 (0.3215 vs. 0.2705). This low precision is understandable as neither dimensional data nor weight measures are good classifiers for our sample data. The weight coefficient analysis computes the coefficient set as [0 0 0 0 1]. This suggests that weight measures are slightly better than dimensional data, and this can be verified by querying Robot Share with only dimensional measures or weight measures, respectively. Results from the last test in this group could be difficult to understand. The test is set as the two most useful features for object classification, image information and text information are corrupted, with one third of the samples estimated by sample means. The query is missing the other image and weight measures. However, with coefficients equal to [0 1 0 0 0], Robot Share is still able to achieve an average retrieval precision very close to its maximum. To understand this, we have to review the basic test setting. The database contains 480 samples, which are derived from 16 real world objects. The way we define the siblings search is as follows: relevant items are items replicated from the same master copy. In this test, one third of the database is corrupted, which means on average 10 copies from every 30 replicated group are corrupted. However, since the ranking tests only analysis results from the first 20 items, Robot Share just happily reports that using edge histograms as the sole classifier, it could find all of the 20 siblings with ease. Hence we again obtained a near perfect score on average precision. When we increase the number of returned items to 30, which equals to the number of siblings one object has, we obtain results displayed in Table 4.17.

Table 4.17. Performance Test 3.3

	Min	Max	Median	Mean	Variance
Coefficient	[0 1 0 0 0]				
Precision	0.4667	0.9000	0.7333	0.7143	0.0095
Recall	0.4667	0.9000	0.7333	0.7143	0.0095
Retrieved	30	30	30	30	0
Coefficient	[1 0 0 0 1]				
Precision	0.6000	0.9000	0.7667	0.7567	0.0062
Recall	0.6000	0.9000	0.7667	0.7567	0.0062
Retrieved	30	30	30	30	0
Coefficient	[1 1 1 1 1]				
Precision	0.1333	0.9000	0.5333	0.5113	0.0239
Recall	0.1333	0.9000	0.5333	0.5113	0.0239
Retrieved	30	30	30	30	0

We observe that the measured precision drops about 30%. We compute weight coefficients under this new condition, and obtain another set of coefficients, which brings a slightly better result. Compared with setting all coefficients to one, the performance gain is more evident.

4.1.4 Supplemental Experiment I

The above experiments have demonstrated Robot Share performance on object data that are collected by ourselves. We have also received a set of object images taken by Prof. Dillmann's humanoid robot research group. These images are used in their textural related recognition research and are quite different from images we collected. Only one image is taken at each object and some images do not capture the entire object. Combining these images with images we found using the Google image search, we have constructed a second database that contains 103 images. We tested Robot Share performance with these images. Experiment results are presented in Table 4.18.

From Table 4.18, we can see the retrieved accuracy is lower than results from previous tests. This is expected as data samples are of lower quality than previous

Table 4.18. Supplemental Experiment 1

Objects	# of Samples	Precision (Top 2 Items)	Precision (Top 3 Items)
Bottle:	7	0.64	0.52
Bowl	15	0.67	0.54
Box:	7	0.79	0.57
Can:	5	0.67	0.72
Chopstick:	8	0.59	0.45
Cup	7	0.57	0.38
Fork	7	0.64	0.47
Jar	12	0.71	0.61
Plate	17	0.76	0.71
Spoon	18	0.61	0.46

experiments, i.e., only one image is record for each object, and no text or any other information is presented. It also reveals the fact that the color, edge histograms based identification is less suitable for randomly collected object samples.

To summarize, in this section, we have examined Robot Share performance from various angles. We have tested how Robot Share performs under optimal conditions. We see that results measured in precisions and recalls are both high. We then have tested queries with incomplete information. Performance drops accordingly. However, the amount of performance degeneration depends on features that are missed. Incomplete records show similar behaviors. Missing certain features in database items are more harmful to the retrieval performance than others. The last group of tests on using weight coefficients to control ranking demonstrates that using the right set of weight coefficients could result in better performance to the system. However, coefficients computations are expensive as this is a multidimensional minimization problems. Coefficients obtained from this procedure are both query and database dependent.

4.1.5 Supplemental Experiment II

We have obtained a set of humanoid robot activity data from Prof. Dillmann's group. Chapter 2 presented the *VooDoo* data representation. These data contain eight activities with each activity performed multiple times. A total of 120 instances of activities is recorded. Since all instances are performed by a human experimenter, recorded lengths of instances range from 41 frames to 151 frames. We randomly select query templates from the activity database. Since the purpose of activity recognition is to identify human activities, Nearest-Neighbor search is more appropriate than k-Nearest-Neighbor search or α-cut search used in previous experiments. We then limit the number of returned activities for each search to be 2 (since every search always returns the query template itself as the first activity.) We define the classification as correct if the second returned item is the same activity as the query template. Results are presented in Table 4.19. Ref. [24] indicates these results are comparable to the FFNS approach used by Prof. Dillmann's group.

Table 4.19. Supplemental Experiment II

Activity	Correct
Hold Out Hand	91.0%
Hold Out Object	95.5%
Put Object On Table	89.9%
Read Book	73.6%
Sitting	89.9%
Standing	86.3%
Take Object From Table	77.7%
Typing On Laptop	100%

CHAPTER 5

FUTURE WORK AND CONCLUSION

Humans started to share knowledge even before the birth of human natural language. Since then, knowledge sharing has played a major role in human civilization. The study of knowledge sharing in man-made intelligent systems, such as robots and software agents, has a relatively young age in the study of artificial intelligence, even though recent developments in Robotic, Semantic Web and Semantic Grid start to touch this topic from various angles. There still is a lot of room for exploration.

Previous chapters presented our work on the design and implementation of Robot Share, a knowledge sharing framework for robots. We have demonstrated how two types of knowledge, object knowledge and activity knowledge, can be shared through this framework. We have discussed reasons behind various method selections and have compared tradeoffs between different designs. The Robot Share system has been examined on a set of different experiment data sources and we have discussed experiment results.

Future research can be divided into two categories: Robot Share refinement and Robot Share expansion. In the refinement department, we propose a set of approaches that are worth trying to discover if any of them gives better search retrieval results. In the expansion department, the concept of a machine-readable knowledge search engine for knowledge sharing can be taken to other domains, such as intelligent software agent. Detailed discussion follows.

5.1 Robot Share Refinement

The current proof-of-concept Robot Share system is centered at image data, even though the framework has been deliberately designed for multiple data formats. Robot Share performance on other types of data such as laser range finder, sonar,

etc. is yet to be examined. To support a new data type in Robot Share, the data type needs to be processed and a vector representation needs to be extracted. Polynomial coefficient representation has been used for existing image data. Additional approaches for dimension reduction need to be evaluated for data from other types of sensors. Similar to dimension reduction methods, distance measures should be evaluated as well. In the current implementation, L1 and weighted L1 distance measures are used. Since the selection of distance measures is tied to the data representation, more distance measures should be evaluated for new data types.

For image based object identification, more object recognition methods should be evaluated. The current approach is based on color and edge histograms. These approaches, especial the edge orientation histograms, are good properties to classify objects, given that a clean background segmentation and textures on objects are not too strong to give false identification on object edges. However, under certain conditions, clean and easily distinguishable edges cannot be found. Even though we have stated that image segmentation is not the problem Robot Share tries to solve, we realize that other object identification techniques, which do not rely on strong edges, can be adopted. Another limitation associated with edge based object identification is, using the edge based object identification, only a broad type of object classification can be obtained rather than fine grained answers. For instance, it is impossible for an edge based object identification system to distinguish a can of Coke from a can of Sprite. However, there are times that the ability to distinguish Coke from Sprite is important. Texture and image pattern based techniques and their associated indexing structures should be investigated.

Feedback systems have been heavily used in today's Internet search engines. We have mentioned that a feedback system could help to adjust better weight measures for object properties in the cross analyzer in Robot Share. We are certain about that feedback process would be a very helpful tool to adjust various parameters in Robot Share. However, the issue we are not so certain about is how feedbacks can be generated by robots. At an abstract level, the merit of a feedback system comes from the robot in field, which sends query to the Robot Share, knowing certain

information that Robot Share does not know. Therefore it is able to evaluate query answers generated by Robot Share better and give useful feedback. This model works perfectly for human users. As the interfaces between human users and computers, especially the input interface, e.g., keyboard and mouse, are highly limited. It is not possible for a user to convey much information to the search engine. However, conveying information between two machines, i.e., a robot and the Robot Share, is much less tedious. Hence it is possible for Robot Share to know all information the querying robot knows, hence the querying robot may not have much additional information to generate useful feedbacks. In this case, the decision for feedback system utilization is more like a burden-shifting between the query robot and Robot Share, i.e., which one takes the responsibility to ensure query responses are properly ranked. We can imagine that there are situations where a query robot cannot give Robot Share all information to rank query responses most properly, due to privacy and other reasons. In these cases, the ability to take feedback from the query robot can be important.

Chapter 3 has mentioned that Robot Share has the potential to discover underlying relations between data from different sensors. For instance, there is a link between the color histograms and the text description of the object, e.g., the word "yellow" implies a certain shape of color histograms. The ability to discover these links can be very helpful as it not only helps Robot Share to estimate missing information in incomplete queries and records, but also provides ground for higher level knowledge abstraction. The method of discovering these underlying relations is a topic yet to be studied.

In the previous chapter, where Robot Share performance was evaluated, we reached the conclusion that certain data properties are more helpful to classify objects than others. In particular, we have found that text descriptions, which are processed through the LSI technique, and edge histograms of objects are better classifiers than dimension and weight measures. It has been noticed that certain properties with low classifying power, e.g., weight measures, when paired with properties with high classifying power, e.g., text description, the performance of

this combination is worse than using text description as the solo classifier. However, what is uncertain to us is the relation, in terms of classifying power, between these properties, i.e., we know that more information does not always yield a better classifier, the pattern of property selection and the resulting classifying power is still unknown. Intuitively, we think certain properties complement others for classifying certain group of objects, and certain properties only correctly classify a subset of object other properties classify. Hence a combination of two complementary properties may result in a much higher classifying power and a combination of two overlapped properties may result in a lower classifying power. However, it is unknown to us that if these relations are object dependent, and if they are impacted by the selected distance measure. Another problem that is directly related to this issue is the cross analyzer. In the current implementation, the cross analyzer aggregates sorted listed from every indexing structure and produces a single list using the weighted L1 distance. Another approach to realizing the function of the cross analyzer is to use a decision tree, i.e., the cross analyzer sequentially examines a list returned by each indexing structure, if an item listed in the list satisfies a certain condition, e.g., within distance α of the query template, and it is also accepted by previous indexing structures, then adds it to the output list. The output list is then sorted. It is unclear to us if this approach is mathematically equivalent to the current implementation.

The previous chapter has presented the Robot Share performance on three sets of data. The second set of experiments, which examines Robot Share with image data collected from the Google image search and Prof. Dillmann's group, shows worse performance than the first set of experiments, which use data collected by ourselves. One of the major differences between these two sets of images is the image taking condition. Existing algorithms can compute the invariance in 2D graphs, i.e., moments are invariant under rotation. Similar algorithms that compute invariants of 2D projections of 3D space objects are yet to be discovered. Even though various kinds of mathematical techniques can be applied to approximate information that is not directly captured in an image, there is a limit on the amount of information

that can be correctly approximated by those techniques. Therefore, it is exceedingly difficult to detect if two images taken at different angles represent the same object. Hence, single image based object identification is limited. We have noticed the recent development in a 3D scanner, which results in 3D scans that are relatively small and low cost and generate 3D model of objects in short amount of time. It will be very interesting to see if algorithms can compare a 3D model of an object to an image to detect if they show the same object. Such algorithms solve a large class of object classification problems for robots. As robots can create databases that contain 3D models for all kind of objects, then robots in the field only need to take images and query such databases for identifications. We human rarely do 2D image to 2D image comparison when we see objects around us. We always compare images, which are projected in our retinas, to some models, which capture much more information than a simple 2D projection, stored in our brains.

5.2 Robot Share Expansion

The previous section discussed various opportunities to improve Robot Share performance. This section discusses the possibility of expanding Robot Share use into other domains. We first explain how a third type of knowledge, scene knowledge, can be added into Robot Share. Then we discuss how the Robot Share architecture can be adopted in the software intelligent agent world.

The idea of scene knowledge arises from the DARPA Urban Challenge project, in which we also participate. The main objective of the DARPA project is to build an autonomous vehicle that runs on urban streets. There are a lot of similarities between an autonomous vehicle and a humanoid robot. For instance, both of the two have sensors and actuators and need to execute in complex real world environment and know a large amount of information to reach an acceptable level of performance. These similarities ensure that an autonomous vehicle also benefits from knowledge sharing frameworks such as Robot Share. However, there are substantial differences between autonomous vehicles and humanoid robots. For an autonomous vehicle, the ability of distinguishing a dinner plate from a dinner

knife is less useful than recognizing a scene of a blocked road or an interaction with a detour sign. Hence we propose the idea that, in addition to the existing object knowledge and activity knowledge, we add a third type of knowledge: scene knowledge into the framework to support queries from autonomous vehicles. Problems needing to be solved include:

- Developing mechanisms to process sensor, especially image, data for the purpose of scene recognition. In the current implementation, image based object identification is based on identifying objects' color and edges, scene identifications require other mechanisms.
- Defining an ontology based semantic description to communicate identified scenes to autonomous vehicles. This helps to define the scene description language between Robot Share and autonomous vehicles.
- Investigating a fast query-response system that supports real-time response to autonomous vehicles queries with high accuracy.

Scene knowledge can be used in other fields beyond autonomous vehicle control, such as emergency control in surveillance systems. Investigating requirements and limitations of using Robot Share supported scene knowledge sharing in those environments is an interesting topic to explore.

In Chapter 1, we stated that robots resemble many characters of software intelligent agents. Agent systems have demonstrated their ability to solve many problems in software engineering. We believe agents also have an unrevealed potential in Grid computing. Grids are large, heterogeneous, and open environment. The size and complexity of such systems suggests that centralized control structures usually fail. Decentralized designs, modularized components, and the ability of localized decision making make agent systems suitable for Grid applications. However, for agents to succeed in Grid, a knowledge sharing framework designed for agents is also needed. Considering the difference between cyberspaces and the real world environment, before we expend the Robot Share into agent world, we need to answer a number of important questions.

- How to define knowledge for agents? For robots, knowledge is defined as: information about objects and activities, in string format, stored in robots' memory. This definition is not suitable for software agents as object information is less useful for them. Activity knowledge may be useful, as if we define activity knowledge as knowledge about carrying out certain computations.

- How to represent knowledge? In the current implementation, ontology based meta data are less consulted for the purpose of indexing. However, for software agents, it could be that the most efficient way to index agent knowledge is through meta data, even though it is difficult to generate high quality meta data for a large amount of information. How to overcome limitations of meta data based indexing is a problem that needs to be solved.

- How to define the communication language for agents? Chapter 1 has presented some research on a semantic web, which aims at developing machine readable web content. How to utilize results from this research to solve problems in Grids is another topic that needs to be explored.

Seeing the potential of intelligent agent systems and similarities between intelligent agents and robots, we believe it is possible to port the Robot Share architecture into agent world and to bring the power of intelligent agents to the next level.

APPENDIX A

DIMENSIONALITY REDUCTION

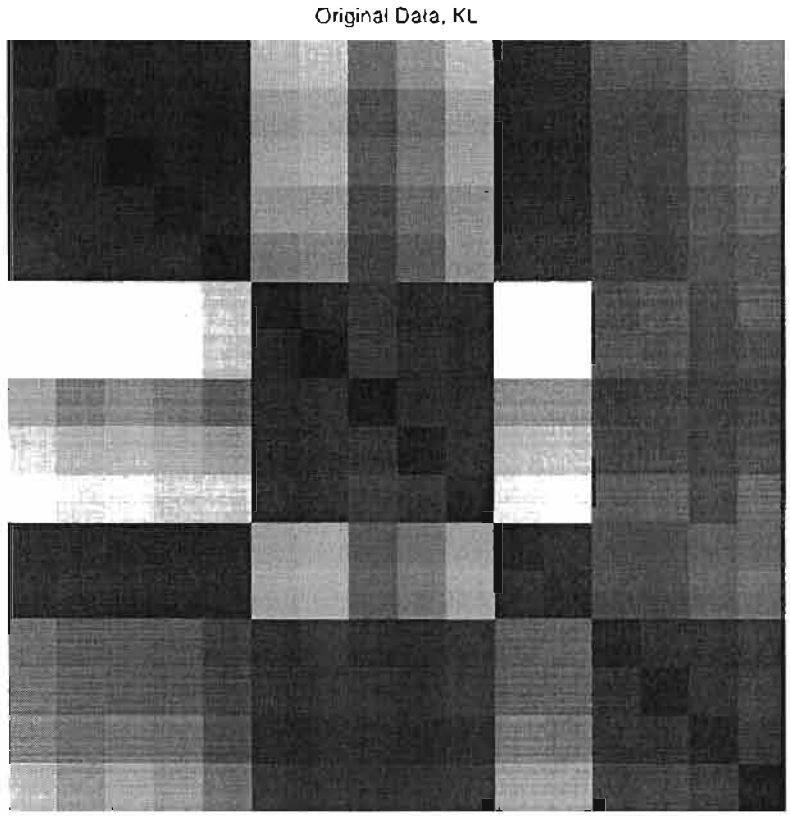

Figure A.1. Confusion Matrix of the Edge Direction Histogram in Kullback-Leibler Divergence

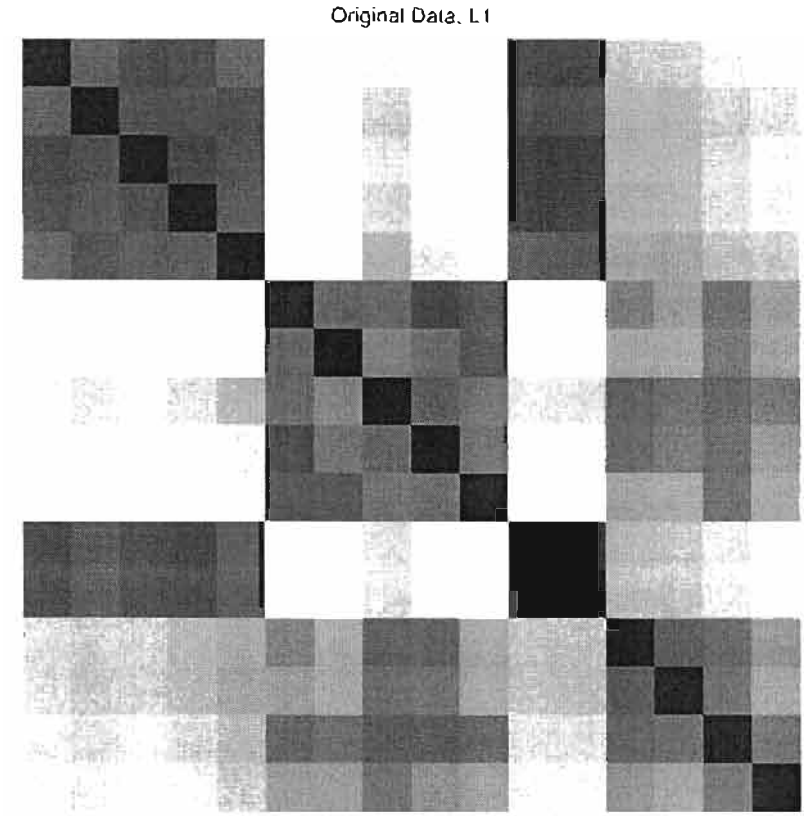

Figure A.2. Confusion Matrix of the Edge Direction Histogram in L1 Distance

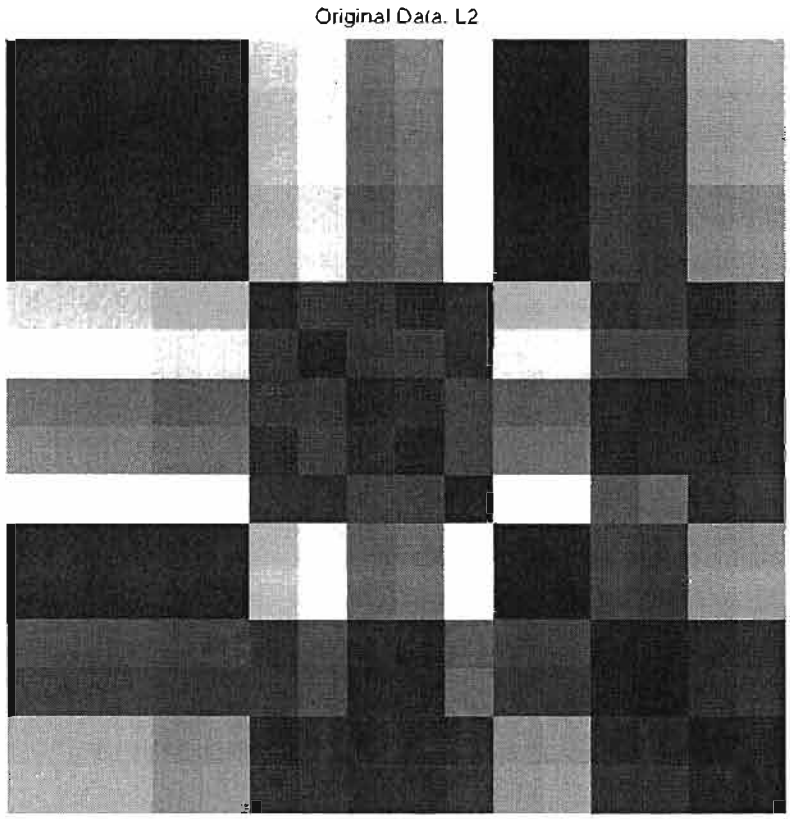

Figure A.3. Confusion Matrix of the Edge Direction Histogram in L2 Distance

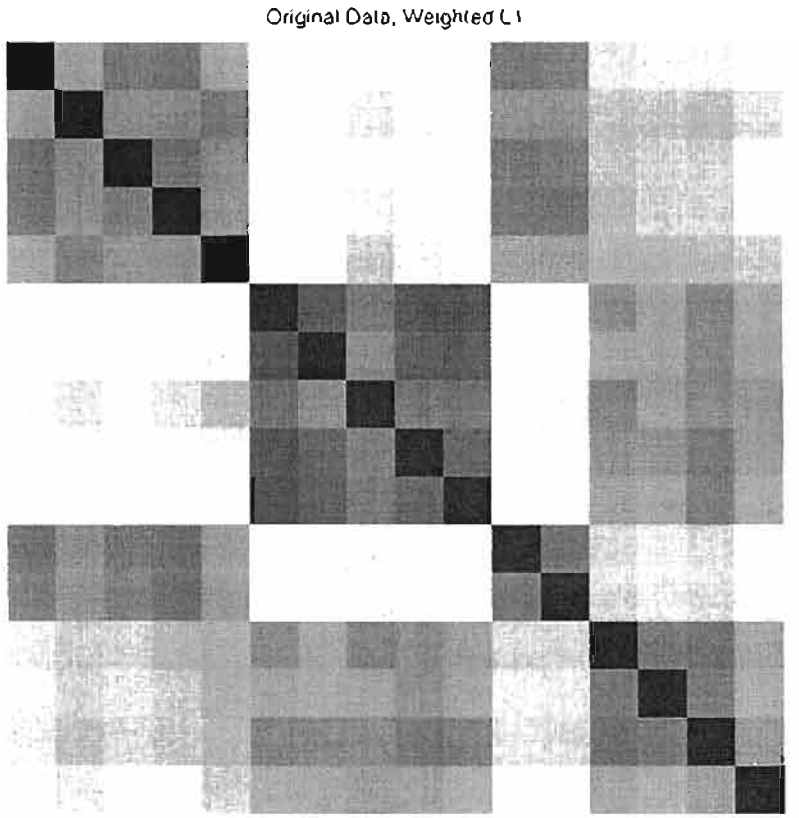

Figure A.4. Confusion Matrix of the Edge Direction Histogram in Weighted L1 Distance

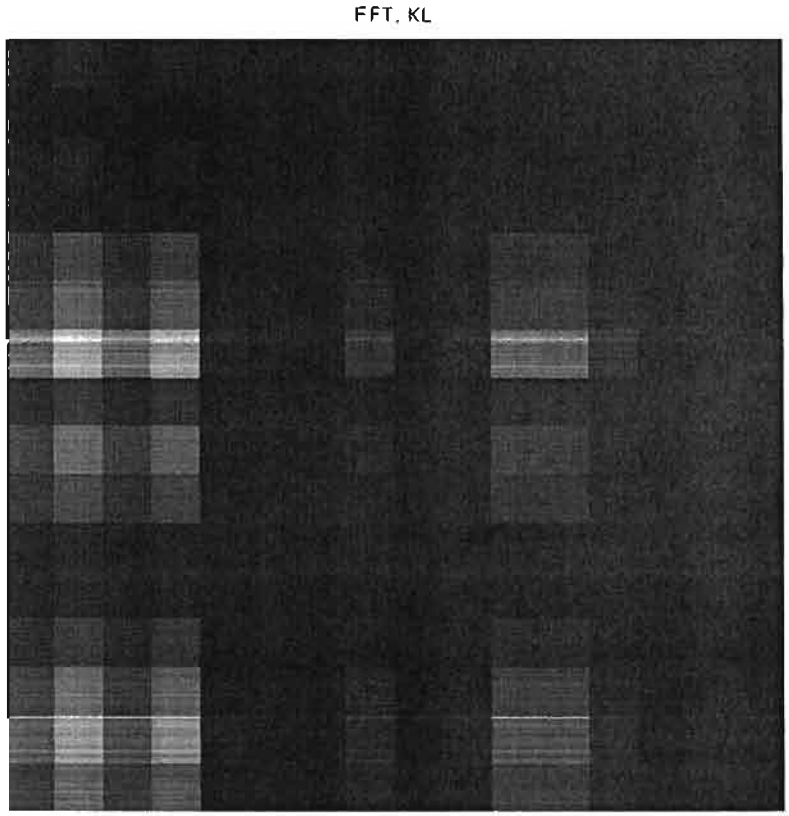

Figure A.5. Confusion Matrix of the Edge Direction Histogram Represented by Fourier Coefficient in Kullback-Leibler Divergence

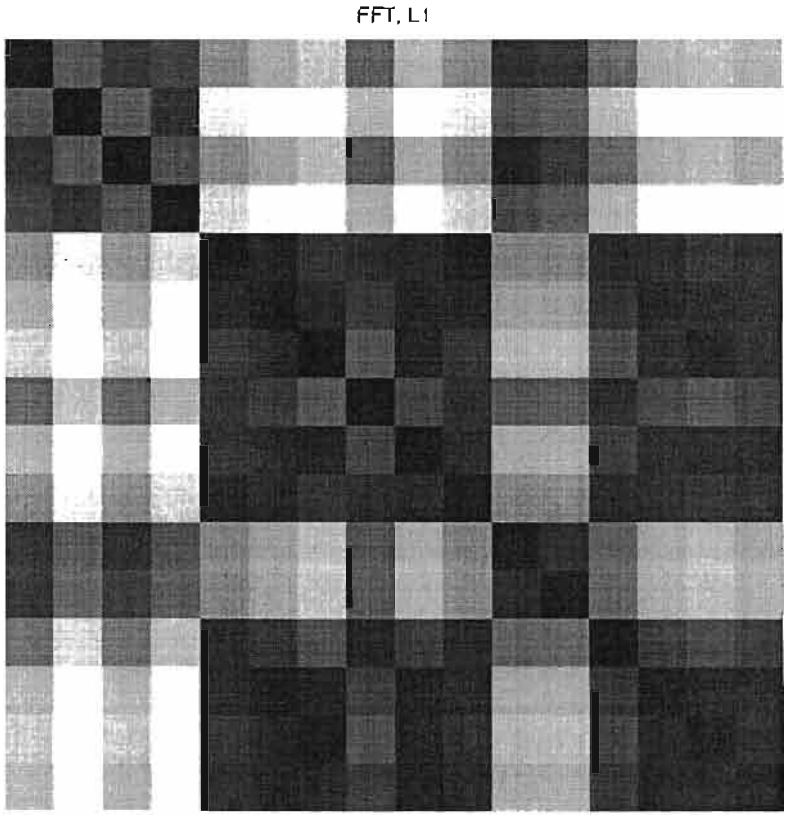

Figure A.6. Confusion Matrix of the Edge Direction Histogram Represented by Fourier Coefficient in L1 Distance

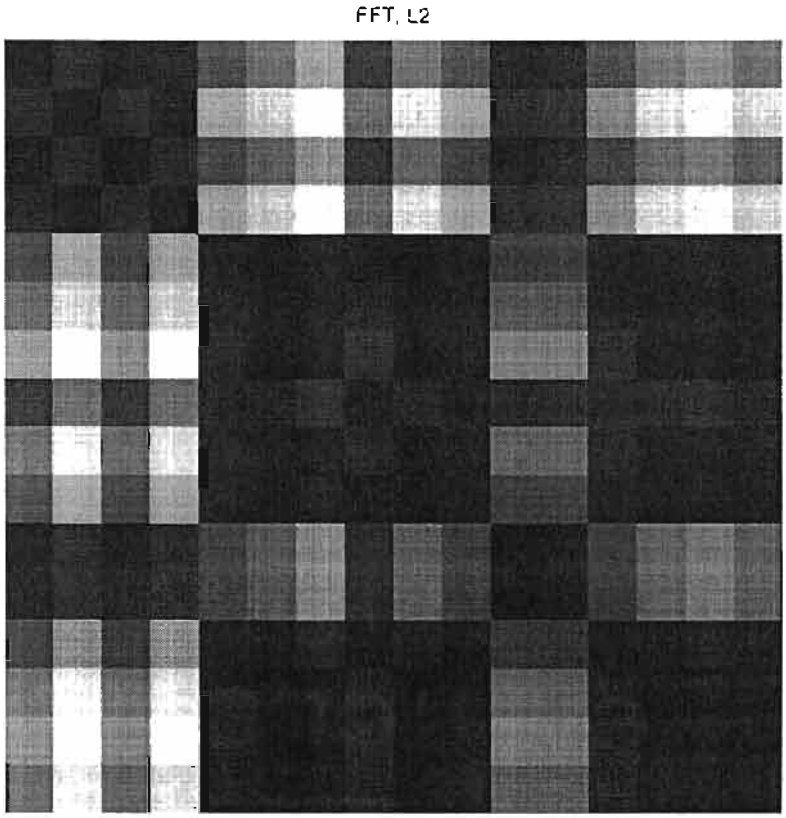

Figure A.7. Confusion Matrix of the Edge Direction Histogram Represented by Fourier Coefficient in L2 Distance

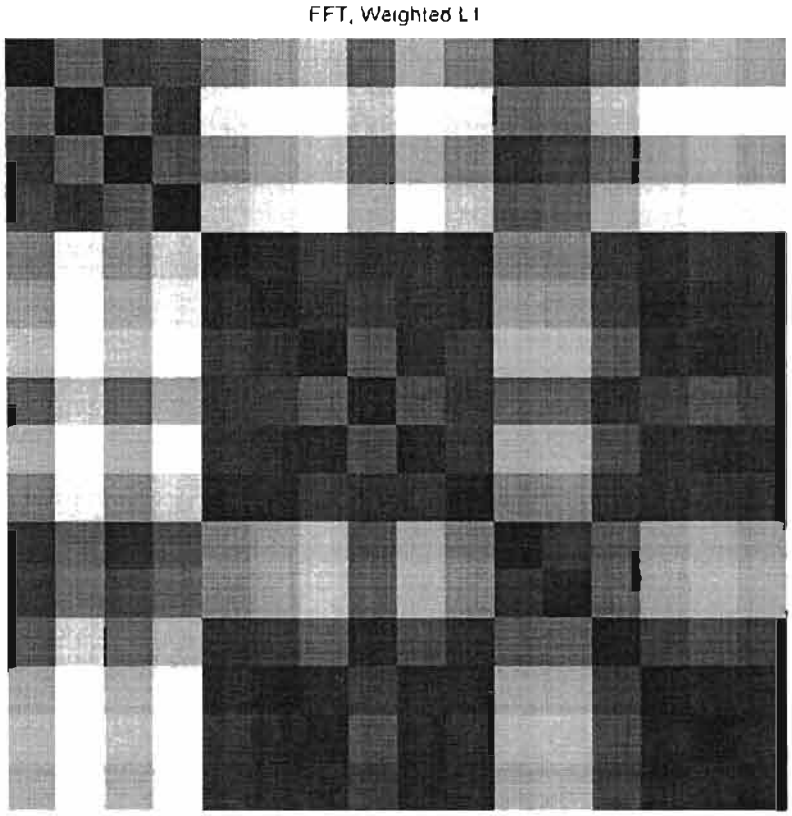

Figure A.8. Confusion Matrix of the Edge Direction Histogram Represented by Fourier Coefficient in Weighted L1 Distance

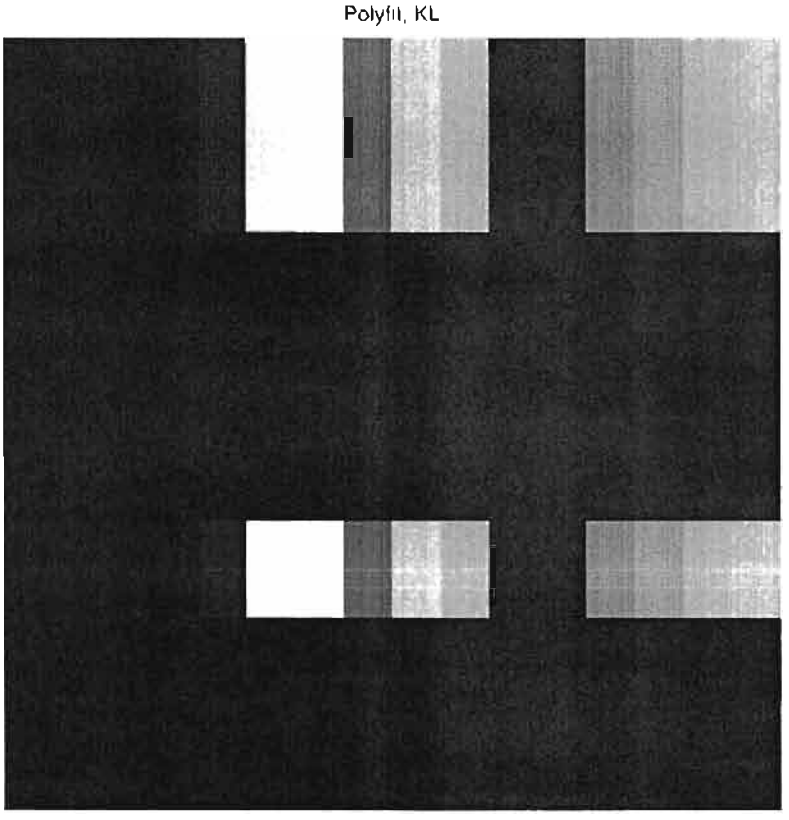

Figure A.9. Confusion Matrix of the Edge Direction Histogram Represented by Coefficients of a Polynomial in Kullback-Leibler Divergence

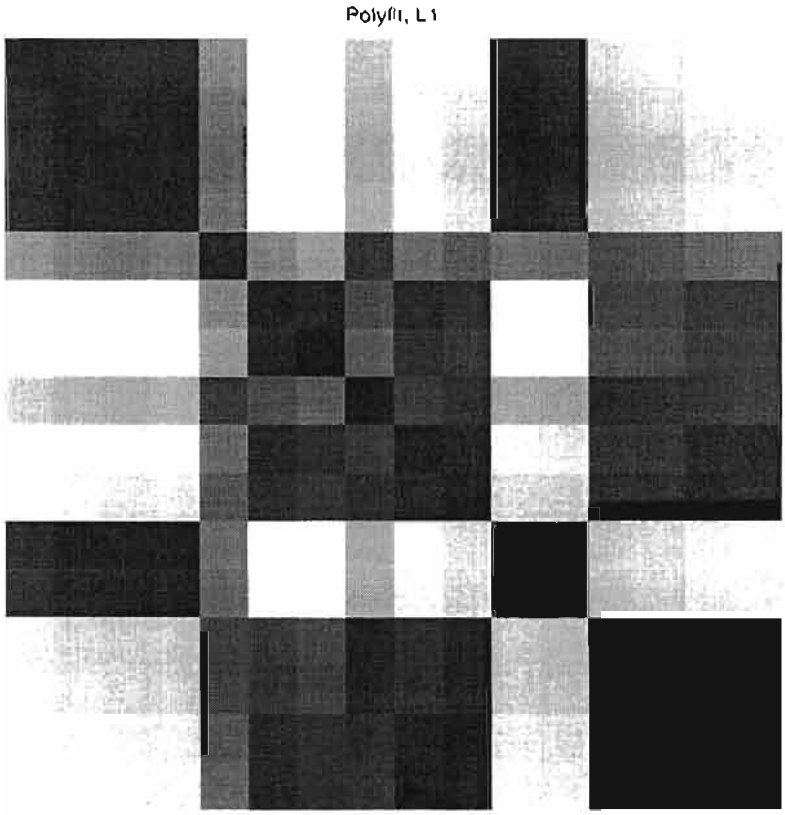

Figure A.10. Confusion Matrix of the Edge Direction Histogram Represented by Coefficients of a Polynomial in L1 Distance

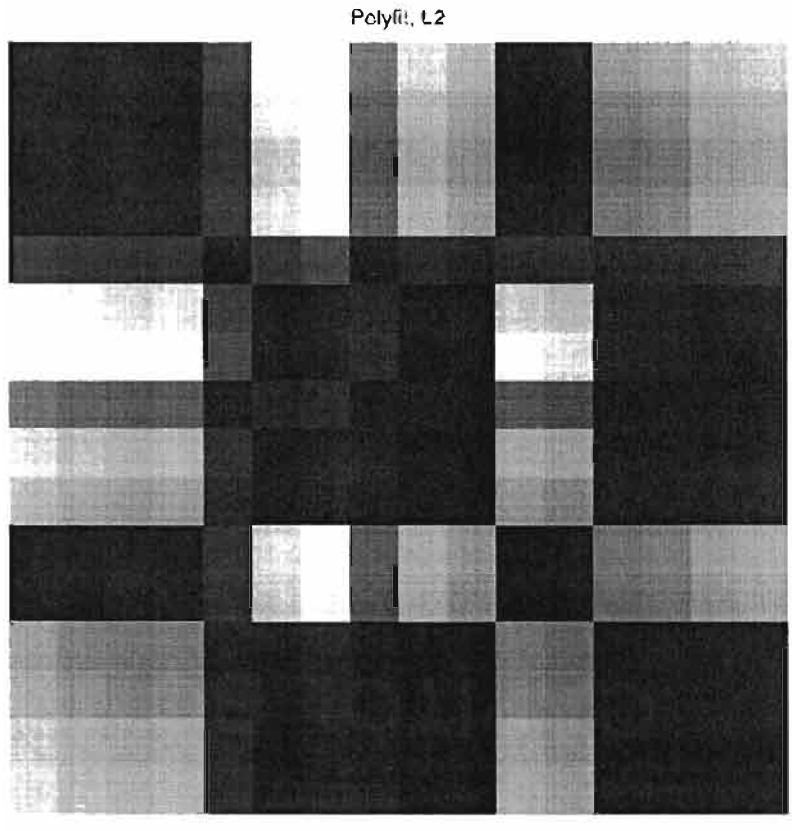

Figure A.11. Confusion Matrix of the Edge Direction Histogram Represented by Coefficients of a Polynomial in L2 Distance

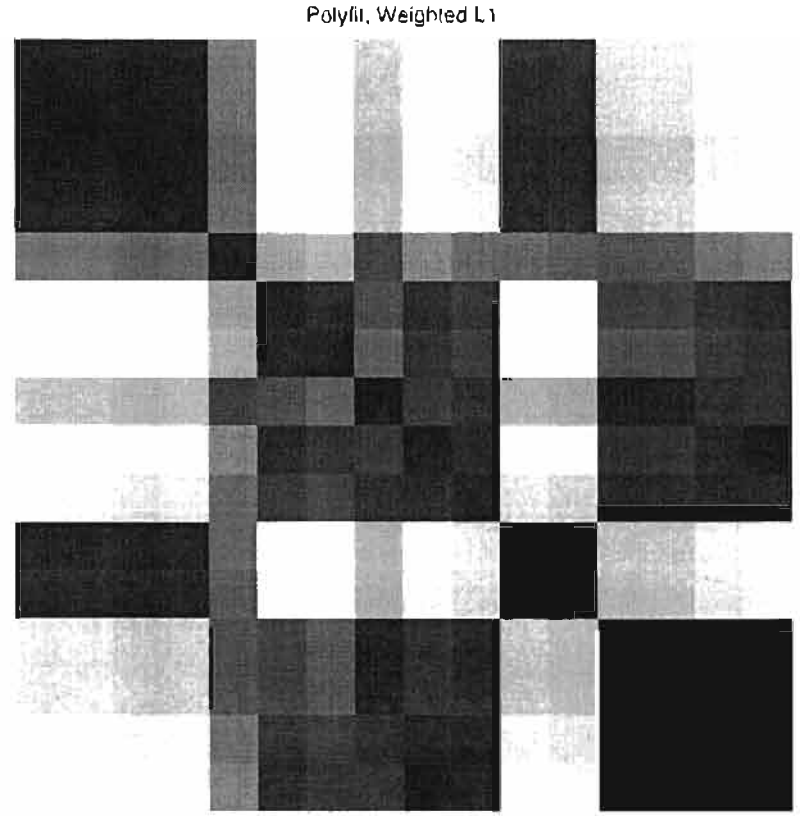

Figure A.12. Confusion Matrix of the Edge Direction Histogram Represented by Coefficients of a Polynomial in Weighted L1 Distance

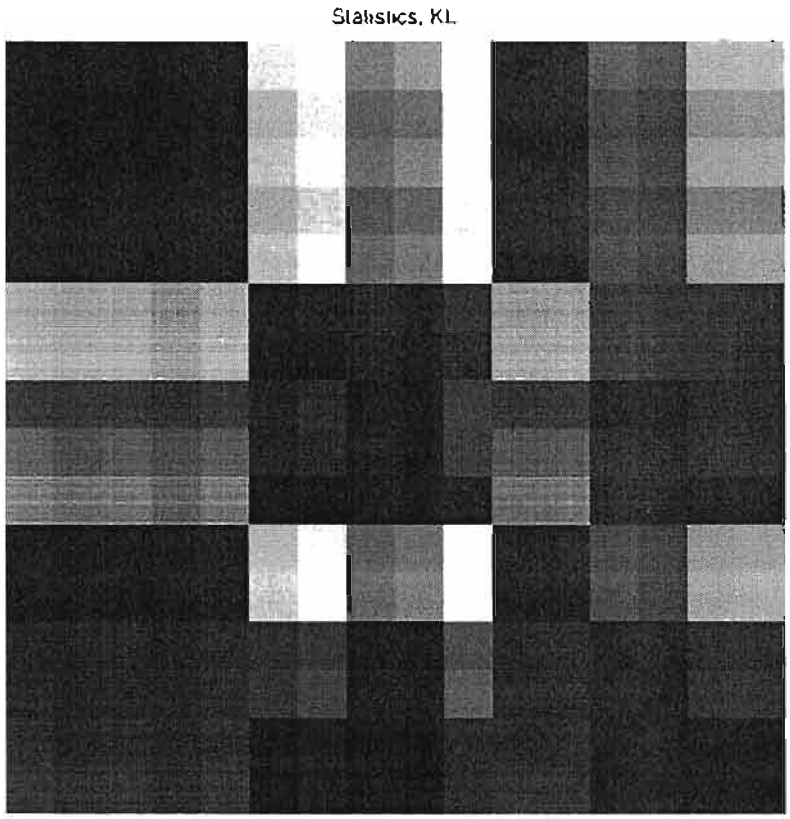

Figure A.13. Confusion Matrix of the Edge Direction Histogram Represented by Statistical Properties in Kullback-Leibler Divergence

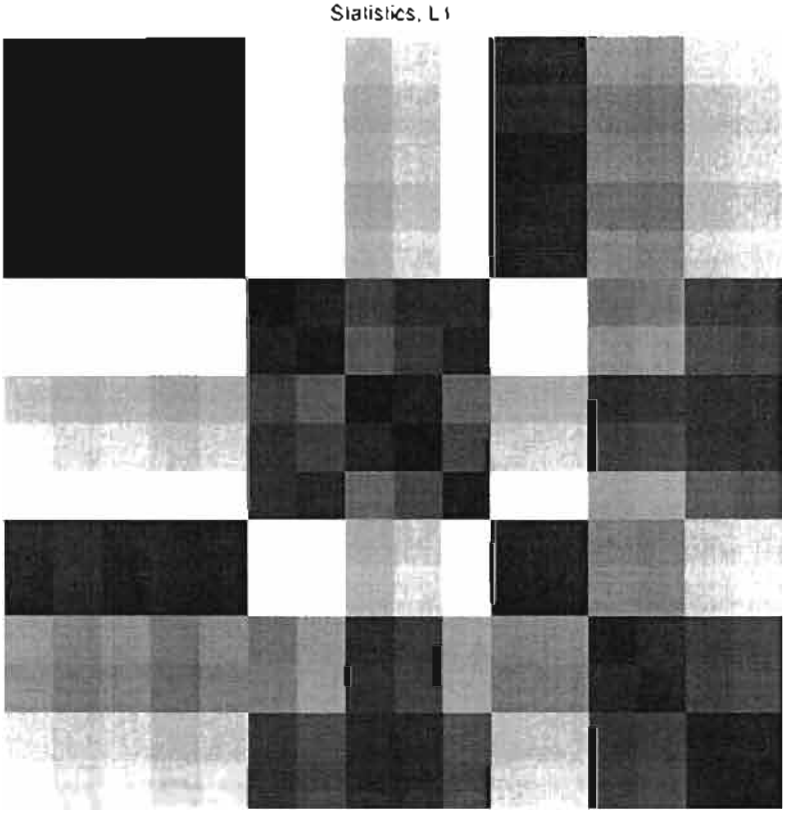

Figure A.14. Confusion Matrix of the Edge Direction Histogram Represented by Statistical Properties in L1 Distance

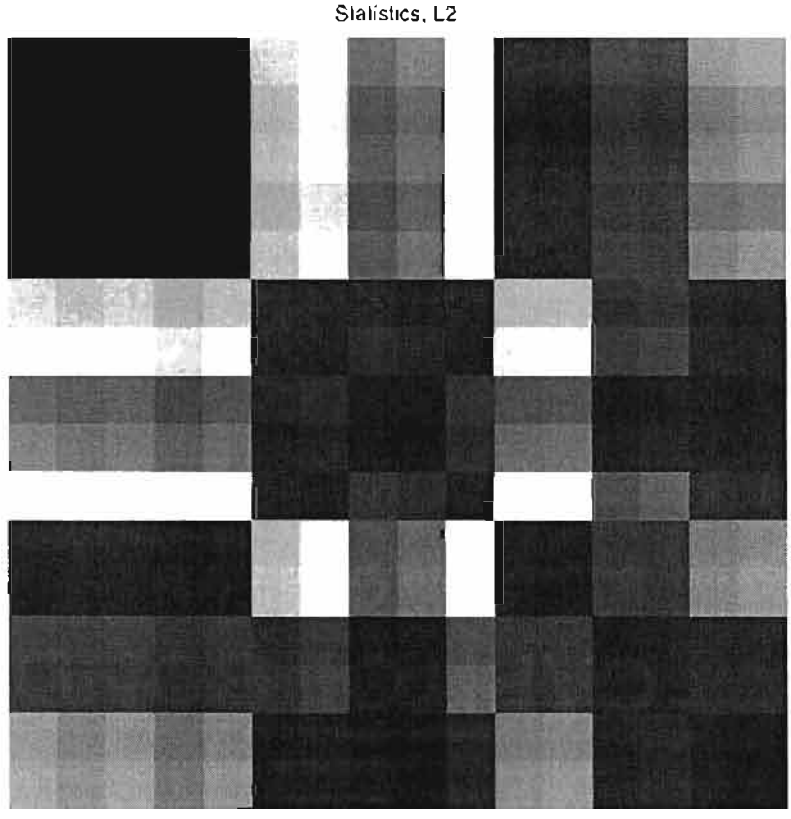

Figure A.15. Confusion Matrix of the Edge Direction Histogram Represented by Statistical Properties in L2 Distance

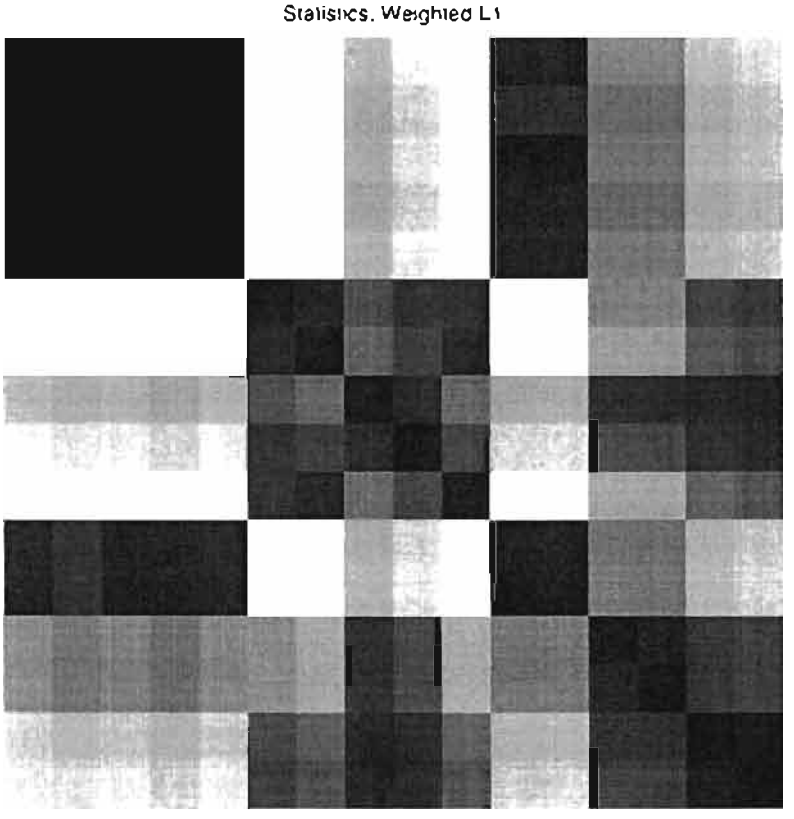

Figure A.16. Confusion Matrix of the Edge Direction Histogram Represented by Statistical Properties in Weighted L1 Distance

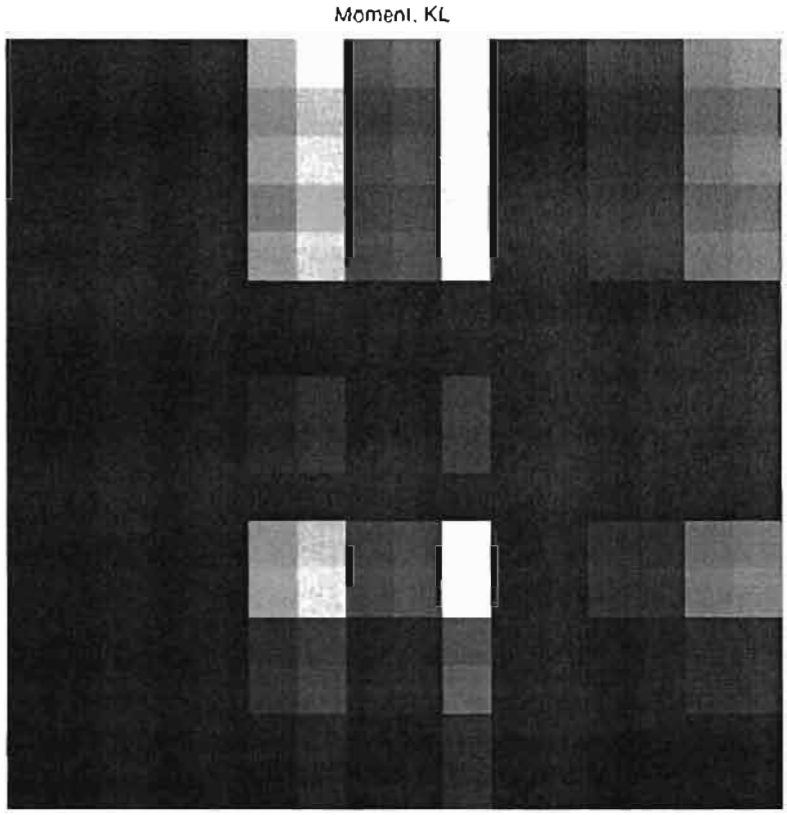

Figure A.17. Confusion Matrix of the Edge Direction Histogram in Represented by Central Moments Kullback-Leibler Divergence

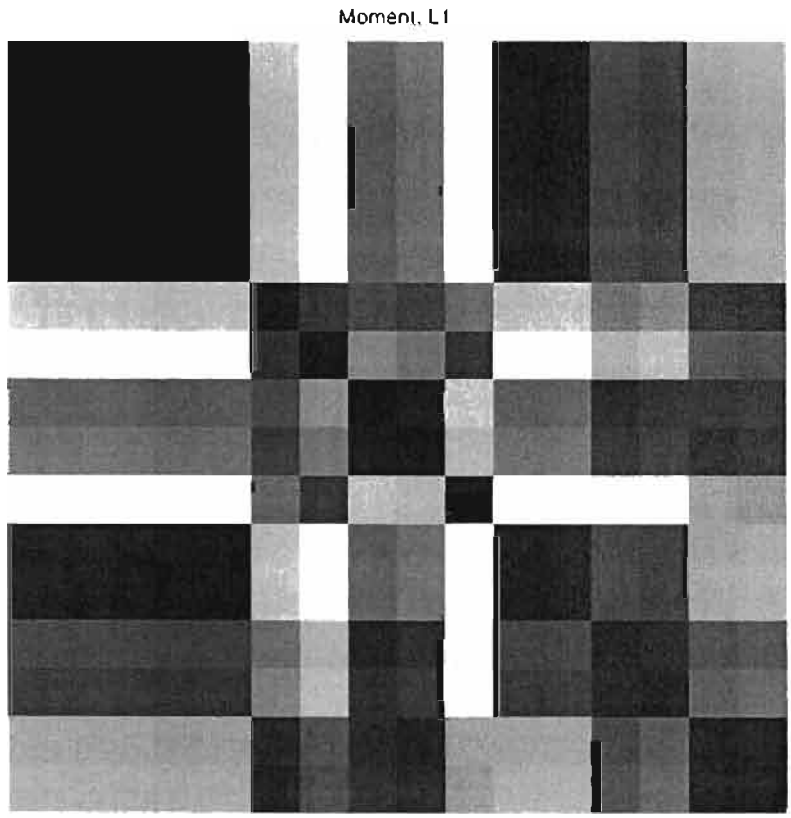

Figure A.18. Confusion Matrix of the Edge Direction Histogram Represented by Central Moments in L1 Distance

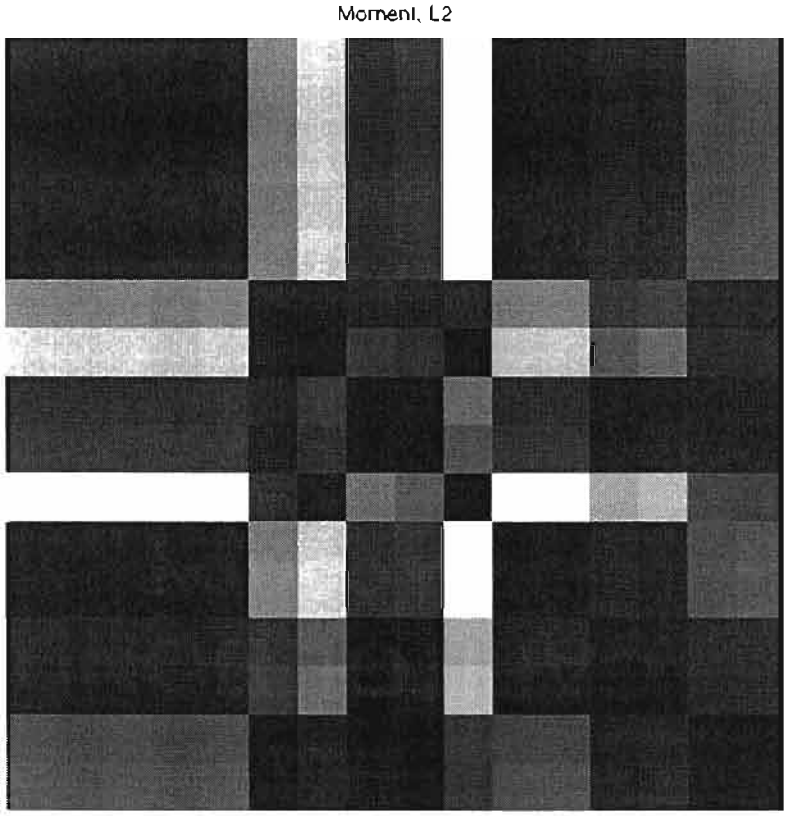

Figure A.19. Confusion Matrix of the Edge Direction Histogram Represented by Central Moments in L2 Distance

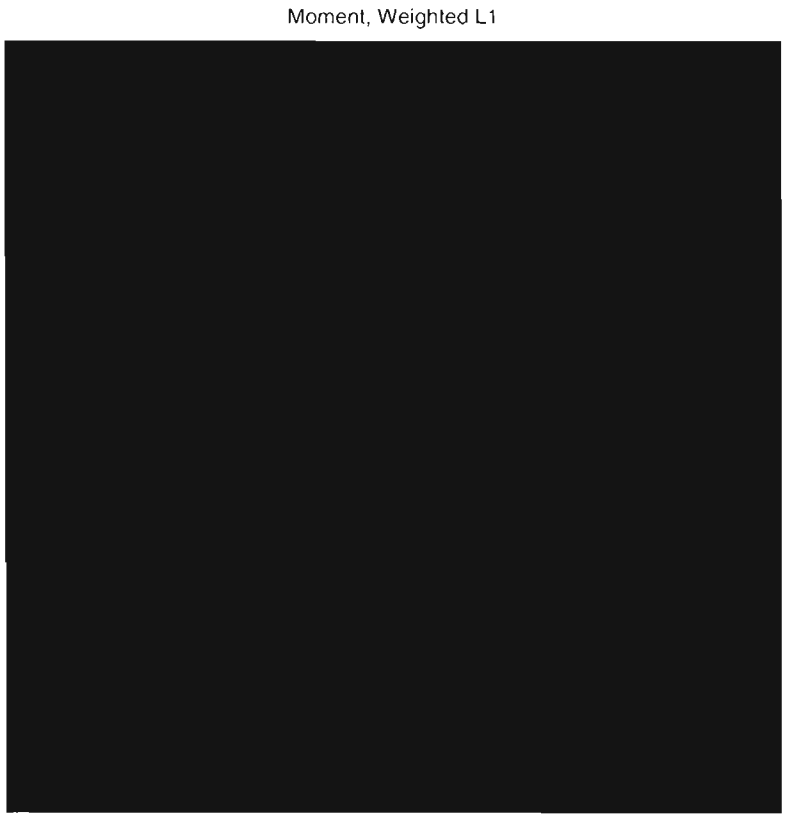

Figure A.20. Confusion Matrix of the Edge Direction Histogram Represented by Central Moments in Weighted L1 Distance.

APPENDIX B

THE KNOWLEDGE DEFINITION GRAMMAR

This grammar describes the definition of object knowledge that is supported by Robot Google.

knowledge

→ ⟨Knowledge⟩ knowledgeBody ⟨\Knowledge⟩

knowledgeBody

→ imageData* dimensionalData textData metaData

imageData

→ fileType image

→

fileType

→ ⟨FileType⟩ type ⟨\FileType⟩

type

→ jpg

→ bmp

image

→ ⟨Image⟩ img ⟨\Image⟩

img

→ STRING

dimensionalData

→ ⟨Dimension⟩ dimension ⟨\Dimension⟩

→

dimensional

→ length width height

length

→ ⟨Length⟩ NUM ⟨\Length⟩

→

width

→ ⟨Width⟩ NUM ⟨\Width⟩

→

height

→ ⟨Height⟩ NUM ⟨\Height⟩

→

textData

→ ⟨Text⟩ text ⟨\Text⟩

→

text

→ name description name

→ ⟨Name⟩ STRING ⟨\Name⟩

→

description

→ ⟨Description⟩ STRING ⟨\Description⟩

→

metaData

→ ⟨MetaData⟩ meta ⟨\MetaData⟩

→

meta

→ time location

time

→ ⟨Time⟩ STRING ⟨\Time⟩

→

location

→ ⟨Location⟩ STRING ⟨\Location⟩

→

STRINGs are printable character strings.

NUMs are positive numerical values.

APPENDIX C

THE QUERY RESPONSE GRAMMAR

This grammar describes the definition of query response sent by Robot Google. The definition of Knowledge is adopted from Appendix B.

response

→ ⟨Response⟩ responseEntry* ⟨\Response⟩

responseEntry

→ knowledge url

→

url

→ ⟨URL⟩ URLSTRING ⟨\URL⟩

URLSTRINGs are regular Uniform Resource Locator strings.

REFERENCES

[1] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepke, and S. Raghavan. Searching the web. Databases and the Web 37, Stanford, December 2002.

[2] P. Azad, T. Asfour, and R. Dillmann. Toward an Unified Representation for Imitation of Human Motion on Humanoids. Unpublished Manuscript, 2006.

[3] R. Becher, P. Steinhaus, R. Zollner, and R. Dillmann. Design and implementations of an interactive object modeling system. In *IRS 2006 37th International Symposium on Robotics*, Munich, 2006.

[4] A. Califano and R. Mohan. Multidimensional indexing for recognizing visual shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(4):373–392, 1994.

[5] V. Castelli and L. Bergman. *Image Databases: Search and Retrieval of Digital Images*. John Wiley and Sons, New York. NY, 2002.

[6] P. Cohen and C. Beal. Natural semantics for a mobile robot. Department of Computer Science 2000-59, University of Massachusetts, 2000.

[7] J. Davies, D. Fensel, and F. V. Harmelen. *Towards the Semantic Web Ontology-driven Knowledge Management*. John Wiley and Sons LTD, West Sussex, England, 2003.

[8] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[9] M. Dekhil and T. C. Henderson. Instrumented logical sensors systems. *International Journal of Robotics Research*, 17(4):402–417, 1998.

[10] R. Dillmann and T. Asfour. Perception, action and cognition through learning of object-action complexes. Technical report, University of Karlsruhe, Germany, Project FP6-2004-IST-4-27657 2005.

[11] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *J. Intell. Inf. Syst.*, 3(3-4):231–262, 1994.

[12] J. Ferber. *Multi-Agent Systems An Introduction to Distributed Artificial Intelligence*. Addison-Wesley, England, 1999.

[13] N. Fielden and L. Kunt. *Search Engine Handbook*. McFarland and Company, Inc., Jefferson, NC, 2002.

[14] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, .Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The qbic system. *Computer*, 28(9):23–32, 1995.

[15] K. Fukunaga. *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA. USA, 1990.

[16] V. Gaede and O. Gunther. Multidimensional access methods. Sonderforschungsbereich 373 1996-69, Humboldt Universitaet Berlin, 1998. available at http://ideas.repec.org/p/wop/humbsf/1996-69.html.

[17] M. Genesereth. Knowledge interchange format. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 389–396, Cambridge, MA, 1991.

[18] R. Grupen and M. Huber. A framework for the development of robot behavior. In *2005 AAAI Spring Symposium Series: Developmental Robotics*, Stanford, CA, March 2005.

[19] S. Hart, R. Grupen, and D. Jensen. A relational representation for procedural task knowledge. In *Proceedings of the AAAI Conference*, Pittsburgh, 2005.

[20] T. Henderson and E. Shilcrat. Logical sensor systems. *Journal of Robotic Systems*, 1(2):169–193, 1984.

[21] T. C. Henderson and R. Grupen. Logical behaviors. *Journal of Robotic Systems*, 7(3):309–336, 1990.

[22] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, NY, USA, 1986.

[23] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):4–37, 2000.

[24] S. Knoop, S. Brannstrom, S. Vacek, and R. Dillmann. Extraction, evaluation and selection of motion features for human activity recognition. In *Proceedings of the International Conference on Robotics and Automation*, Rome, Italy, 2007.

[25] S. Knoop, S. Vacek, and R. Dillmann. Sensor fusion for 3d human body tracking with an articulated 3d body model. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, Florida, 2006.

[26] B. Kuipers, P. Beeson, J. Modayil, and J. Provost. Learning from experience in the ssh. In *AAAI Spring Symposium Series: Learning Grounded Representations*, Stanford, CA, March 2001.

[27] M. Lew, editor. *Principles of Visual Information Retrieval*. Springer-Verlag, London, UK, 2001.

[28] C.-S. Li and V. Castelli. Deriving texture feature set for content based retrieval of satellite image database. In *ICIP '97. Proceedings of the 1997 International Conference on Image Processing (ICIP '97) 3 Volume Set-Volume 1*, page 576, Washington, DC, USA, 1997. IEEE Computer Society.

[29] P. Maes. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*. The MIT Press, Cambridge, MA, 1990.

[30] J. Puzicha, Y. Rubner, C. Tomasi, and J. M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. In *ICCV (2)*, pages 1165–1172, 1999.

[31] J. T. Robinson. The k-d-b-tree: a search structure for large multidimensional dynamic indexes. In *Special Interest Group on Management Of Data SIGMOD '81: Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data*, pages 10–18, New York, NY, USA, 1981. ACM Press.

[32] D. Roy. Grounding words in perception and action: computational insights. *Trends in Cognitive Sciences*, 9(8):389–396, 2005.

[33] D. Roy. Semiotic schemas: a framework for grounding language in action and perception. *Artif. Intell.*, 167(1-2):170–205, 2005.

[34] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, Upper Saddle River, NJ, 2003.

[35] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Boston, MA, USA, 1990.

[36] S. Sclaroff, M. L. Cascia, S. Sethi, and L. Taycher. Unifying textual and visual cues for content-based image retrieval on the world wide web. *Computer Vision and Image Understanding*, 75(1-2):86–98, July 1999.

[37] J. F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[38] M. Stricker and M. Swain. The capacity of color histogram indexing. In *CVPR94*, pages 704–708, 1994.

[39] V. S. Subrahmanian, P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Ozcan, and R. Ross. *Heterogeneous Agent Systems*. The MIT Press, Cambridge, MA, 2000.

[40] G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, MA, 1999.

[41] Wikipedia. Information retrieval — wikipedia, the free encyclopedia, 2007.

[42] Wikipedia. Knowledge — wikipedia, the free encyclopedia, 2007.

[43] M. Wooldridge. *Reasoning About Rational Agents*. The MIT Press, Cambridge, MA, 2000.