

Spatial Reasoning for Robots:
A Qualitative Approach

by

Alan Frank Blackwell

Thesis submitted in fulfilment of the
requirements for the degree of
Master of Science in Computer Science.

Victoria University of Wellington

October 1988

Abstract

This thesis presents a new approach to the problem of shape representation for high level robot reasoning tasks. Techniques from the field of qualitative physics are combined with current methods for solid modelling to develop a qualitative representation of two dimensional shape and position. The qualitative spatial representation has been used to solve simple spatial reasoning tasks. The results of this investigation are applicable both to the field of robotics, where they provide a new approach to programming and control, and to the field of qualitative physics, which has been hampered by a lack of general purpose spatial reasoning techniques.

Acknowledgements

Thanks to Peter Andrae, my supervisor, for his fund of ideas and his energetic criticism. Also to PROGENI, for the use of office facilities, and for extended periods of study leave. Thanks for advice, assistance, and friendship to Mark, Karen, Duncan, Bernd, Paul, Craig, Julie, Jonathan, Matthew, Andrew, Mike, David, Evan, Philip, Alvin, Alex, Clifton and Steve. Thanks to my musical friends for a life beyond study, especially Harry and Gillian. Lastly, thanks to Helen. This project would never have commenced, progressed, or been completed without her encouragement and support.

Contents

1	Introduction	1
1.1	Limitations of Robotic Spatial Reasoning	1
1.2	Spatial Reasoning in Qualitative Physics	2
1.3	Advantages of Spatial Reasoning to Qualitative Physics	3
1.4	Advantages of Qualitative Reasoning to Robotics	4
1.5	Qualitative Spatial Reasoning Scenarios	6
1.6	Organisation of the Thesis	8
2	Spatial Representation and Reasoning in Robotics	10
2.1	Survey of Robot Reasoning in AI	11
2.2	Space and Shape Representation for Robots	14
2.2.1	Robot Motion Representation	15
2.2.2	Space Filling Representations	16
2.2.3	Shape Boundary Representation	20
2.2.4	Constructive Solid Geometry Representation	21
2.2.5	Object Relative Position Representation	22
2.2.6	Representation for Robot Vision	23
2.2.7	Other Robot Representations	24
2.3	Shape Representation in Other Fields	25
2.3.1	Wireframes	26
2.3.2	Boundary Representation	26
2.3.3	Constructive Solid Geometry	27
2.3.4	Other Methods	28
3	Qualitative Reasoning in Spatial Domains	29
3.1	The Origins of Qualitative Reasoning	29
3.1.1	Early Qualitative Reasoning Systems	30
3.1.2	Naive Physics	31

3.2	Techniques in Qualitative Reasoning	33
3.2.1	Classification of Qualitative Reasoning Tasks	33
3.2.2	Elements of Qualitative Reasoning	35
3.3	Spatial Qualitative Reasoning	37
3.3.1	The Bouncing Ball World	37
3.3.2	The Mechanism World	39
3.3.3	Other Spatial Reasoning Systems	41
4	Two Methods for Qualitative Representation of Shape and Space	43
4.1	Representation Issues in Qualitative Robot Reasoning	44
4.1.1	Representation of Detail	45
4.1.2	Geometric Reasoning Using Local Contexts	47
4.1.3	Representation of Multiple Features Using Functional Groupings	49
4.1.4	Qualitative Representation of Size	50
4.1.5	Summary of Qualitative Representation Issues	53
4.2	2D Qualitative Geometry from Solid Modelling	54
4.2.1	A Two Dimensional Derivative of Constructive Solid Geometry	55
4.2.2	A Two Dimensional Derivative of Boundary Representation	56
4.3	Qualitative Two Dimensional Shape Description	57
4.3.1	Describing 2D Shape with Axially Specified Subparts and Features	58
4.3.2	Describing 2D Shape With Extended Polygon Boundaries	65
4.4	Extensions to the Two Dimensional Qualitative Representations	75
4.4.1	Including Order of Magnitude Information in the Distance Ordering	75
4.4.2	Ordering of Angle Sizes	76
4.4.3	Explicit Links to Three Dimensional Shape	78
5	Implementations of Two Qualitative Spatial Reasoning Systems	79
5.1	A Program for Reasoning About Sliding	80
5.1.1	Representing Contact State	81
5.1.2	Contact State Transitions	86
5.1.3	Envisionment of Motion through Contact States	89

5.1.4	The ASSF Implementation	90
5.1.5	Summary of Sliding Issues	92
5.2	Reasoning About Path-Planning with EPB/PDO	94
5.2.1	The Path-Planning Problem	94
5.2.2	EPB/PDO Implementation ¹	95
5.2.3	Stages in Path Planning	97
5.2.4	Gap Finding	99
5.2.5	Checking for Fit	100
5.2.6	Specifying Motion	103
5.2.7	Directional Reasoning	103
5.2.8	Summary of Path-Planning Issues	105
5.3	Future Enhancements	107
5.3.1	Integration of two approaches	107
5.3.2	Searching for Complex Paths	107
5.3.3	Unfastening Problems	108
6	Conclusions	110
6.1	Evaluating Spatial Qualitative Reasoning	110
6.1.1	Using PDO/EPB in Domains from Other Projects	111
6.1.2	General Evaluation of a Qualitative Representation	113
6.2	Evaluating Qualitative Robot Reasoning	116
6.2.1	Reasoning with Incomplete Information	117
6.2.2	Providing Graceful Degradation	118
6.2.3	A Human Interface for Robot Programming	119
6.2.4	Robot Reasoning with PDO/EPB	120
6.3	Summary	121
6.4	Future Research Directions	122
A	EPB/PDO Representation Example	124
B	Robot Fastening and Disassembly Projects	150

List of Figures

1.1	Scene Example for the “Mover’s Problem”	7
2.1	Space Filling Representations	19
3.1	Free Space Division Guided by Scene Features	38
4.1	Proximity Measurement in Terms of Boundary Expansion . .	71
4.2	Apparent Object Interpenetration at Coarse Levels of Detail	72
5.1	Vertex to Vertex Contact	81
5.2	Line Segment to Vertex Contact	82
5.3	Aligned Line Segments Contact	82
5.4	Overlapping Line Segments Contact	83
5.5	Large Line Segment to Small Line Segment Contact	83
5.6	Vertex to Convex Curve Contact	84
5.7	Vertex to Concave Curve Contact	84
5.8	Line Segment to Convex Curve Contact	84
5.9	Line Segment to Concave Curve Contact	85
5.10	Convex Curve to Convex Curve Contact	85
5.11	Concave Curve to Convex Curve Contact	86
5.12	Implementation of the EPB/PDO Representation	98
A.1	Scene for EPB/PDO Example (drawn to scale)	125

Chapter 1

Introduction

The subject of this thesis, qualitative spatial reasoning, falls between two areas of artificial intelligence research: qualitative reasoning, and robotics. The work described draws on past research in both of these fields, and has resulted in the development of a new technique for qualitative representation of shape and space, together with methods for performing simple spatial reasoning tasks using the representation. I intend that the results presented here should be useful in finding new approaches both to robot planning and problem solving, and to qualitative reasoning, while being of interest to a more general audience insofar as they represent a plausible computer model for certain aspects of human spatial reasoning.

1.1 Limitations of Robotic Spatial Reasoning

Robots are general purpose machines. Current robotics technology makes them mechanically capable of performing a wide variety of operations. In principle, it is possible to reconfigure a single robot so that it can be used for a range of different tasks; in practice, most commercial users of robots find it uneconomic to reconfigure robots in any situations which cannot be simply demonstrated by “lead-through” methods, because of the programming expenses involved. The problem with robots is that although they are general purpose machines, it is very expensive to instruct them for complex tasks.

Commercial robots can perform tasks only when they have been given a precise and detailed set of instructions for carrying out those tasks. The greatest challenge for current robotics research is to implement, in a com-

puter system, the kind of “intelligence” that will enable robots to be more easily instructable. In human terms, robots need to be less stupid [Pug82]. The ways in which they are particularly stupid include the inability to act on goal oriented instructions, to plan sequences of actions, to learn from their mistakes, or to understand the world around them; international research in robotics has been addressing these problems since the 1960’s.

Research toward providing intelligence for robots is motivated both by the long term commercial benefits of more intelligent robots [Bla86], and by the challenges of robotics for those involved in the wider field of Artificial Intelligence research. Those challenges result from the fact that robotics requires the interface of a computer system to the physical world by way of sensors and actuators. Much early A.I. research dealt either with domains which were purely synthetic (such as computer programming, or chessboards), or with “toy worlds”, which are simplified computer models of very restricted real world situations.

Whereas robotics imitates the behaviour of intelligent animals by performing the intelligent connection of perception to action, synthetic problems involve no perception or action, and are thereby simplified to a point at which the real problems of robotics disappear. Toy world problems can involve real perception, but act on such a simplified version of the world that many real world problems do not occur. This avoidance of the real world has been criticised by Brady [Bra85a], and by Hayes [Hay83], who suggest that future artificial intelligence research should be linked to problems in robotics.

Brooks [Bro86] has proposed robotics as a starting point for A.I. research, for evolutionary reasons. He points out that higher intellectual functions have only developed in animals over the last few thousand years, once the ability to act in the physical world was well established. He argues that an understanding of the real world as a participant in it is a necessary prerequisite to the kind of “common sense” which current A.I. research finds great difficulty in achieving.

1.2 Spatial Reasoning in Qualitative Physics

Qualitative physics is a field of research that provides a new repertoire of reasoning techniques for computers. These techniques enable computers to reason about the real world without using numerical information or methods. The aims of qualitative physics are related to those of “Naive Physics”,

which attempts to describe the world using a formalised version of human non-specialist understanding, thereby developing a model of human reasoning and also providing commonsense knowledge for A.I. systems. Qualitative physics tends to be applied to more restricted problem domains than naive physics – particularly engineering problem domains such as hydraulics, formal mechanics and electronics, rather than to problems in the everyday world.

Qualitative physics research to date has concentrated on problems that do not involve much spatial information, dealing instead with complex processes or device interaction over time. The analysis of systems such as circuits can be performed by using a simple representation of the connections between components in the circuit, rather than any spatial description of the circuit components.

Early qualitative reasoning systems operated in spatial problem domains, but used methods of reducing the spatial content in the problem until the programs operated only on connectivity information. The ways in which this was achieved are described in detail in chapter 3.

1.3 Advantages of Spatial Reasoning to Qualitative Physics

Most problem domains in physics involve a certain amount of spatial content, but qualitative physics research has been restricted to problems with minimal spatial content. This has greatly restricted the utility of qualitative physics methods, because few ways have been developed of applying successful techniques to a more general range of problems. This weakness has been noted by a number of people working with qualitative reasoning systems [Fal87] [FNF87] [Jos87].

Recent qualitative reasoning literature describes systems which reason about the behaviour of mechanisms, rather than circuits. In these situations, shape and space information is important, and the development of qualitative spatial reasoning methods has been identified as a priority. The techniques described in this thesis are an alternative approach to spatial reasoning that is more general than that currently used for mechanism analysis.

1.4 Advantages of Qualitative Reasoning to Robotics

Current robot planning systems operate on a precise geometric and numeric description of the robot, its workspace, and the objects that it is manipulating. Programmable robots (as opposed to guided, or “lead-through” robots) are programmed to perform motions according to this precise description, which seldom includes any tolerance information. Adequate precision of the robot in carrying out its program, and exact location of objects in the workspace is therefore a major concern of industrial robotics, because sufficient precision must be obtained so that the numeric description agrees with the workspace.

Despite impressive achievements of precision in robots, there are many problems which cannot be solved by simply increasing precision. It is these problems which can benefit from the use of qualitative spatial reasoning methods instead of numerical geometry. A few important problems for advanced robot programming that may be solvable by the application of qualitative methods are as follows:

- A robot will encounter errors in its data from time to time, regardless of normal operating precision. In these situations it should be able to continue operating, perhaps less efficiently, rather than having to abandon the task. This is known as “graceful degradation” in performance.
- High level robot programming should be carried out without having to refer to numeric data. Ideally, a robot programmer should describe the task to the robot in terms that they would use to describe it when doing it themselves (“task-level” programming). People do not naturally think of physical actions in terms of joint angles or numeric workspace co-ordinates, so high level robot programming should be done in non-numeric terms.
- A common way for people to communicate information about spatial tasks is by the use of diagrams. A diagram normally reflects the structure of the task only, without containing any important dimensions in the actual lines of the diagram – it is a qualitative device. A robot should be able to understand the structure of the task from this kind of information.
- Where parts of a robot’s workpiece are hidden, and the dimensions are therefore completely unknown, the robot must be able to make

hypotheses about their shape. Consider the removal of a key from a keyhole, for instance – we must make some mental picture of what the hidden part of the key might look like, before developing a strategy for removing it. If numerical information is needed for robot operations, an hypothesis must include a complete geometric description, whereas a qualitative reasoning system allows the robot to proceed on simple structural hypotheses.

- If the task involves design, it is often necessary to propose hypothetical values for some design parameters (where the design is under-constrained). In this case, the hypothetical assumptions made should include no more information than is absolutely necessary to continue design, since added information may cause the hypothesis to fail unnecessarily. The use of a qualitative representation can provide this facility, by describing only essential structural features.

An example of a current project in high level robot reasoning which could benefit from the use of qualitative representations is Andreae's NODDY system [And85], which forms the basis of an ongoing project at Victoria University. This system observes the actions of a conceptual robot in a geometric world, and learns about robot procedures by generalising from its observations. Numeric information is largely irrelevant to acquiring the functional aspects of useful procedures, although NODDY presently uses a number of numerical techniques. A qualitative description of robot actions would provide a set of information for generalisation to proceed from that had already been filtered to isolate structural elements.

There are a wide range of robot reasoning tasks which are hampered by the complexity of operations in three dimensional numerical geometry. The original starting point of my research was a proposal to investigate the understanding of fasteners through robot disassembly of real mechanical devices. The proposal excluded the physical issues of vision and manipulation, but still resulted in a list of almost 100 research areas which would need substantial progress before a start could be made on the complete disassembly problem.¹

Many of these disassembly and fastening research issues derived their complexity from the fact that representations available in three dimensional robotic reasoning systems were not appropriate to the task that I wished to

¹The relationships between these research topics is illustrated in the diagrams of appendix B.

solve. This failing originated the investigation of qualitative spatial reasoning that is described in this thesis.

1.5 Qualitative Spatial Reasoning Scenarios

This section summarises two tasks which are typical robot reasoning problems, and which a useful qualitative spatial reasoning system should be able to perform. The system described later has successfully performed both of these tasks.

The first task deals with sliding motion in two dimensions:

Given a scene consisting of a collection of objects, and one object designated as the moving object, what contacts can be established between the moving object and the others, under the constraint that during all motion, the moving object must remain in contact with at least one other (i.e. it is sliding around the other objects).

The problem here is to produce what is called an “envisonment” in qualitative physics – a set of possible future states for a system which is in a known initial state. The major difference between a “robot” problem such as this, and a qualitative physics problem, is that the actions of a system analysed in qualitative physics normally result from instability in the initial state, whereas here the robot has to physically act on the system to produce a change of state.

Finding a solution to this problem requires firstly that the object boundaries and position be represented in such a way that possible sliding directions can be found, and secondly that the results of any motion can be described in this representation, so that further motions can be planned. The three reasoning stages in finding a problem solution are therefore:

1. Propose a possible slide from the current position.
2. Determine the possible contact positions at the first change of “state” after this slide.
3. Compile a history of states that would be achieved after a series of such sliding motions.

The second task is a simple two dimensional case of the “Mover’s Problem” or the “Findpath” problem, which can be stated as:

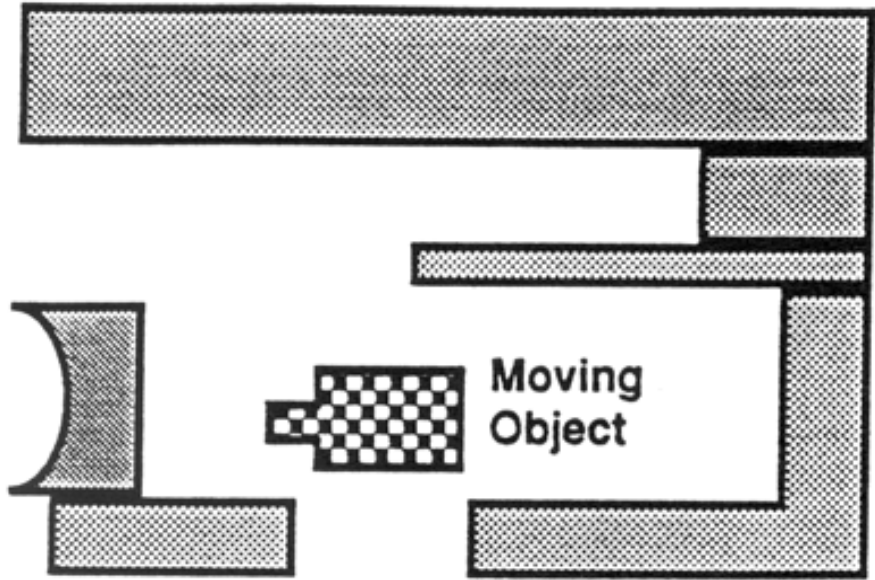


Figure 1.1: Scene Example for the “Mover’s Problem”

Given an initial scene arrangement consisting of an object to be moved, and N obstacles to the motion; find a path for the object through the obstacles, such that it will not collide with any of them.

Note that the mover’s problem is considerably more complex in three dimensions than it is in two. I have discussed in chapter 4 my reasons for limiting my approach to the two dimensional case. A two-dimensional scene is shown in figure 1.1.

The problem for the robot is to find a path for the moving object which will move it from amongst the surrounding obstacles, so that it can be either removed from the scene, or manipulated in open space.

In solving this problem, the reasoning system must ask the following questions:

- What obstacles prevent the moving object from moving into free space?

- Are there any gaps between those obstacles which might provide a path?
- Are any of the gaps large enough for the object to fit through?
- What direction should the object be moved in, if it is to approach a suitable gap?
- Are there any other obstacles along the path to that gap?
- If there are other obstacles, how should the path be modified in order to avoid them?

This thesis describes a qualitative representation for physical objects and scenes. The representation has been used to describe workspaces for the two tasks presented above. Algorithms for carrying out the tasks have been implemented, and both the algorithms and their implementations will be discussed.

1.6 Organisation of the Thesis

Because this thesis draws from the fields of robotics and qualitative reasoning, it starts with two survey chapters. Chapter 2 presents an overview of research in high level reasoning for robots, and discusses the spatial representations which have been used in several projects. Chapter 3 surveys the brief history of research in qualitative reasoning, covering both representation and reasoning methods, together with problem domains. This chapter includes a critique of qualitative reasoning research with respect to spatial reasoning performance.

Chapter 4 discusses two original qualitative shape and space representation methods, including the “Extended Polygon Boundary/Partial Distance Ordering” representation (EPB/PDO). Each method has been developed using a conventional solid modelling paradigm as a starting point. The methods are discussed in terms of the requirements of simple spatial operations. Chapter 5 presents algorithms which use these representations to perform the tasks described above, with some discussion of the implementation that has been carried out. Finally, chapter 6 discusses the utility of the EPB/PDO representation when applied to qualitative reasoning and high level robot reasoning problems. Topics for further investigation in both qualitative reasoning and robot planning are described.

Appendix A gives a full description, in the LISP language, of a scene used for testing the mover's problem system. This example gives some idea of the implementation issues that arise in providing computerised qualitative reasoning, and can be used as a reference for those with further interest in the details of the representation developed in this project.

Chapter 2

Spatial Representation and Reasoning in Robotics

A fundamental step in the development of an AI system (or any computer system) is designing appropriate computer representations for entities in the subject domain, for the properties of those entities, and for relationships between those entities. (The entities may be physical objects, computer models of objects, or abstract facts). The resulting representation then has a profound effect on the structure, operation, and capabilities of the whole system. This is particularly noticeable in the field of qualitative physics, because the distinctive nature of the field is largely in the novel representation used, and a completely new approach to computer reasoning has developed as a result of that representation.

The application of qualitative reasoning techniques in a spatial domain depends primarily on the development of a (qualitative) representation for physical objects and spatial relationships between objects.

Techniques for the representation of physical objects using computer systems have been developed in the context of several different fields. This chapter surveys the field of robotics, pointing out the approaches that have been taken to spatial representation and reasoning in the past. For comparison, the final section of the chapter briefly surveys shape representation methods used in the areas of computer graphics, and computer aided design. Some of the representation methods that have been used in these fields can be adapted to make a basis for qualitative representations, and this will be discussed further in chapter 4.

2.1 Survey of Robot Reasoning in AI

Early AI projects in robotics included Minsky’s “hand-eye” project at the Massachusetts Institute of Technology in 1966, McCarthy’s “computer with hands eyes and ears” at Stanford University in 1968, and Rosen and Nilsson’s “intelligent automaton” at the Stanford Research Institute in 1967. As one can deduce from the anthropomorphic titles of these projects, their overall aims were for computers to imitate human capabilities when acting in the real world, as a complement to the reasoning powers that were being developed under the auspices of AI research.

These projects were mainly concerned with investigating the technology required for a robot actuator guided by sensory data, but the techniques developed were soon applied to practical tasks – in particular the task of mechanical assembly. Robot assembly projects were soon under way at Stanford (assembling a water pump) in 1973, University of Edinburgh (assembling a toy car) in 1975, and Hitachi Central Research Lab (assembling objects from plan drawings) in 1972 [EUY⁺72].

These hand-eye assembly projects only had a primitive internal representation of the task they were performing. For example, the Edinburgh project, described in [ABB⁺75], used a camera looking from above onto a pile of white parts on a black table, and identified white areas which protruded from the general pile. A grasp at one of these positions would take hold of a single object. This object could then be pulled out of the pile, laying it flat on a clear area of the table, so that its shape could be matched against a known library of parts. Each identified part was then stored in a predefined place, so that the assembly subsystem (which used no visual information), could follow a set sequence of movements to incorporate it in the assembly.

A number of early AI programs operated on a problem domain where a robot arm was building piles of blocks on a tabletop. This problem domain became known as the “blocks world”. Many of the programs that operated on the blocks world were not interfaced to an actual robot, or to sensing equipment. In these cases, the blocks world was simply a model in computer memory, which included only the information necessary to solve the questions that the program would be applied to. One example of such a blocks world program is the well-known SHRDLU, by Winograd, which responded to English language commands, and answered questions about the effects of its actions on the world model.

“Blocks world” programs usually investigated reasoning capabilities which

were seen as being useful to real robots. Fahlman's BUILD [Fah73] analysed the stability of piles of blocks, and planned ways in which it could build and dismantle piles without them falling over. Sussman's HACKER [Sus75] investigated machine learning by comparing the results of action plans to goals that it had formulated for those plans; it could acquire new "skills" by storing the debugged versions of successful plans.

Since these early projects robotics research has diversified, and now takes place at various levels of abstraction from the physical robot. Typical research topics (in order of increasing abstraction) are:

1. New manipulator, arm, and sensor designs.
2. Control of manipulators and arms, and the interpretation of sensors.
3. Description and specification of robot motion.
4. Automated motion planning.
5. Operation from task descriptions, rather than motion description.
6. Deriving assembly plans from a simple description of the assembly.

New manipulator developments include general purpose devices such as multi-fingered manipulators (e.g. [JWKB84] [LPD83]), as well as special purpose manipulators for handling unusual objects ([KTTP83] [BH82]). Arm designers experiment with different numbers of joints and different coordinate systems (spherical, cylindrical, and cartesian). New sensors include tactile and contact sensors, together with remote sensors such as sonar or optical devices (e.g. [Sie86] [RT82] [BDdRP83] [Ben83]).

Design of new actuators or sensors must consider control issues, but research continues even on the control of well-known devices. Robot arm kinematics is constantly under analysis, and the control of dextrous hands appears to be at least as complex as the design of the hand itself. The interpretation of visual images is also a very large field of research, although the methods of acquiring the images have not changed very much for 20 years.

The specification of robot motion is essential for programming robots, and a variety of techniques have been developed for describing such motion. These range from robot programming languages [GSCT83], to methods for guiding or leading the robot through a desired trajectory. All of them require

methods for analysing and describing the motion of a robot arm (e.g. [Lyo85] [TPB81]).

If a robot is to plan motions for itself, it must be able to plan and evaluate possible paths. A considerable body of work has been devoted, for example, to the problem of collision avoidance for an arm moving in three dimensions [Bro83]. Motion planning issues also include compliance [LP85] – controlling the arm so that it exerts a specified force at the surface of a workpiece – and grasp analysis for manipulators [Ngu85b] [Lyo85].

An attractive goal for robot users is the ability to describe only the task that must be performed, while the robot works out for itself the details of the motions that it must carry out. This goal is called “task-level programming”, as opposed to the “robot-level programming” that describes individual motions. Task-level programming languages include Lozano-Perez’s LAMA [LP79] [LP76] and AUTOPASS by Lieberman and Wesley [LW77]. These languages describe tasks in terms of the workpiece, rather than in terms of the robot.

“Goal-level programming” aims to have the robot carry out even more of the work necessary in deciding how to perform a task. A typical scenario in goal-level programming is that the programmer describes an assembly, then the robot works out for itself how to combine available parts to form the assembly. General discussions of goal-level robot programming are given by MacDonald [Mac87a] and Zhang [xZ87].

Industrial robot applications have mainly made use of developments from the first three levels of abstraction in the above list. The later three levels have only been investigated in a research context, and experimental task-level or goal-level systems often operate in a simulated “blocks world”, rather than using real robots or workpieces. The integration of robot control across these varying levels of abstraction is an important research topic. One promising approach to this topic is hierarchical integration, as proposed by Brooks [Bro85b] and Albus [Alb81].

Further progress in the more abstract levels of robot reasoning relies largely on the development of more sophisticated representations of the physical world, as has been pointed out by Brady [Bra85a]. The reasoning tasks carried out in high level robot control tend to involve qualitative rather than numerical data, as the reasoning moves further from exact motion description (this was noted by Ambler and Popplestone in developing the RAPT system for assembly goal specification [APK82] [PAB80] [AP75]). The qualitative representation for spatial reasoning described in this thesis developed from an investigation into qualitative, high-level representations

for use by robots.

2.2 Space and Shape Representation for Robots

All robot systems operate in the physical world, and a robot controller therefore must include some form of spatial representation, whether it is an explicit description of object shapes and locations, or whether it is implicit in a sequence of motions through space. The requirements of robot spatial representations vary considerably in different applications, however – the sophistication of spatial representations must increase as the robot controller is required to perform more sophisticated tasks. The following list presents the types of spatial representation facilities that are necessary to carry out different levels of robot programming and control:

- Robot level programming does not assume that the robot has any explicit knowledge about its surroundings. The shape of the objects that the robot is operating on is, however, *implicit* (to some extent) in the motions that the robot makes.
- Vision systems (those which perform object recognition, or scene analysis) must have some explicit spatial representation, since all visual processing operates on a two dimensional projection of the physical subject.
- “Hand-eye” systems must be able to relate the description of an object as it appears in the visual field to operations that the robot will carry out in its workspace. This requires a spatial representation which is more versatile than for vision alone.
- Systems in which object descriptions are directly created by a programmer require that the description can be readily derived from the programmer’s own “spatial representation” – his concepts of space and shape. At the same time, the robot controller must be capable of relating the programmer’s description both to features in the visual field, and to actual movements. Representation facilities like these are a requirement of task level programming.
- Where a robot is reasoning for itself in an environment that may contain unknown objects, or unfamiliar arrangements of objects, it must be able to use sensory data to construct a useful spatial representation

of its surroundings, so that it can plan and operate in those surroundings.

- “Robots” that operate in a simulated world are of course particularly dependent on the form in which the simulated world is represented, since they do not interact with anything other than the representation.

The remainder of this section discusses several approaches which have been taken to space and shape representation in robotic systems. It includes discussion of techniques that have been used in most of the above categories of task, so that the capabilities of various general methods can be compared.

2.2.1 Robot Motion Representation

The great majority of present day commercial robots are programmed using “robot-level” programming technology. The only spatial representation necessary to support this type of programming is a stored description of robot motion. Typical components of such a description are:

- Joint angles for the robot arm
- Workspace coordinates
- Forces which the robot must apply
- Velocity of the arm (either angular, or cartesian)

These components are often amalgamated in a “trajectory”, which describes the sequence of positions, velocities, or forces that the programmer has specified. A sophisticated program may have a number of trajectories, which can be selected or modified, depending on sensory input.

A powerful set of techniques for motion representation are the configuration space methods, which describe robot motion in a multi-dimensional space. This is useful in describing general robot motion, where it can be necessary to simultaneously move an object in any of three directions (x,y, and z axes), while rotating it around any of three axes. The complete motion can be planned and described in a six dimensional configuration space. Path planning techniques using this type of configuration space are described by Lozano-Perez [LP83], Gouzenes [Gou86], and Donald [Don87].

Multi-dimensional motion spaces for reasoning about force control problems are described by Mason [Mas81]. His system specifies motion along

“C-surfaces”, which are surfaces in a 6-dimensional space that defines force application in three directions, in addition to position. C-surfaces can be used to describe the motion involved in complex force control tasks such as drawing on a blackboard with chalk, or tightening a screw while holding a screwdriver in the slot.

Although the shape and arrangement of workpieces are to some degree implicit in the programmed trajectory or C-space motion, it would be difficult to derive very much useful information about the workpiece from this “representation”. (There is certainly some simple information that could be deduced – consider, for example, the possibility of finding the body shape of a car from the path followed by a spray painting robot).

I include a discussion of these motion representations for completeness, but they are really too specialised (to robots) to be applied to more general questions of spatial reasoning.

2.2.2 Space Filling Representations

In a space filling representation, an area of space (either the whole workspace, the visual field, or simply the locality of an operation) is partitioned into regions, which are classified as either occupied or unoccupied. An object is represented in the system simply as a collection of occupied regions. The spaces between objects are represented as explicitly as the objects themselves, and can be analysed by considering the collection of unoccupied regions.

The technique used in the Edinburgh assembly project described in the previous section is typical of the most primitive form of space-filling representation. In this system the basic units of space in the representation correspond exactly to pixels in the robot’s visual field. The Edinburgh system made two dimensional images using an overhead camera. The two dimensional outline of an object lying on the table was then matched against a library of outlines for all parts. The identification of protrusions from the pile, and outline shape matching could both be carried out using a simple copy of the visual image as the internal representation. This copy was a pixel map with bits of an array set where there were light areas in the image.

The same simple representation was used in this project as the basis for visually guided arm movements. Each stored object representation had associated with it a point by which the object could be grasped, and once the identity and position of the object had been established relative to the camera (which was in a fixed position), the arm could be guided directly to

the grasp point.

This hand-eye system is versatile enough to operate in many industrial environments, but the simple representation of shape results in several limitations. One limitation is that shape matching is carried out on the whole object, and where an object is partially obscured, the system cannot recognise it. The practical solution to this problem involved using the robot arm to scatter piles of objects which obscured each other; this would not be an acceptable technique in a lightbulb assembly plant! A further limitation is the restriction to parts that are adequately described by a two dimensional view.

The Edinburgh system was limited to recognising shapes that have unique outlines when lying flat on a table; it is possible, however, to use the same simple level of visual shape encoding, with more sophisticated indexing, to obtain a more complex description of three dimensional shape. The simplest way of doing this is to store outlines of an object when viewed from different angles. Grossman and Blasgen developed such a system for identifying objects, which was not limited to flat parts [GB75]. Complex parts were vibrated in a tray into one of a few stable orientations. The vision system was then able to test the image of a part in the tray for all known orientations.

The representation of the object in this case consisted of a collection of pixel maps of the possible two dimensional projections of the part in each stable orientation. Although this system was able to identify three dimensional parts more accurately and reliably than a simple overhead view, the resulting collection of projections was really only useful for part identification – a robot would require more information about the underlying part structure to plan manipulation.

A limitation of both of the above systems is that the world representation contains only the shapes that are in the visual field of the robot. Space filling representations can be made more powerful by representing the whole of the robot's workspace, rather than just its visual field. This representation must be less dependent on sensing hardware, since it cannot make direct use of image pixels.

The extension of space filling representation to the whole workspace can also involve an extended number of dimensions in the representation. If the workspace representation is not restricted to construction from the information in a single image, it can include three dimensions. Three dimensional space filling divides the workspace into regions using a three-dimensional grid, where the elements of the grid are occupied or unoccupied "voxels".

Space filling representations (whether two dimensional or three dimen-

sional) can be made more compact and memory efficient by using schemes to describe contiguous chunks of either occupied or unoccupied space. One simple method is that of “quadtrees” (or “octrees” in the three dimensional case), which partitions the workspace using the following algorithm: The workspace is divided into quadrants (or octants). If a quadrant is homogeneously filled or empty, it is not subdivided any further. A non-homogeneous quadrant is itself divided into quadrants, and the process is repeated. This subdivision can then be carried out to any level of detail.

Lozano-Pérez’s original LAMA task-level programming system [LP76] makes use of a still more sophisticated space filling approach for representing the overall workspace. This does not divide the workspace into arbitrary quadrants, but into object-specific domains at the boundaries of objects. This method can achieve improvements in accuracy, because the edges of objects are located precisely, instead of being represented at the nearest line of area units. It also involves less memory usage than the quadtree method, because the number of space divisions increases only with the number of objects (or edges), rather than with resolution accuracy. The method is, however, more computationally complex.

Figure 2.1 illustrates the various space filling methods, as used to describe a simple scene at low resolution.

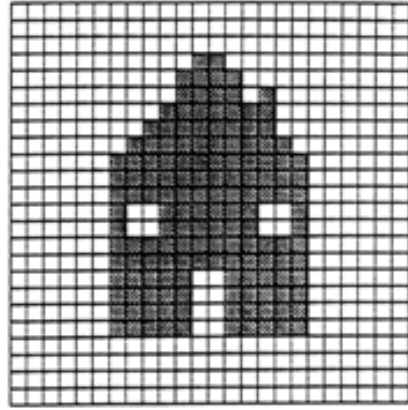
There are two very significant limitations to the space filling techniques. The first of these is that the level of accuracy which can be achieved in describing the object boundary is limited to the size of the smallest unit region. An n -times increase in accuracy therefore results in a 2^n increase in memory usage for a two dimensional pixel-based system, and a larger relative increase for a quadtree system.

The second limitation is that straight-sided objects only appear to have simple boundaries if those boundaries are aligned to the grid along which the representation space is divided. The representation of an object with edges which are not aligned to the grid axes results in aliasing of the edges (this can be seen in the diagonal lines of the example). The efficiency of using regions which fit the object boundaries is also greatly reduced in this case, because it is impossible to subdivide the workspace orthogonally to fit edges. This is noted by Lozano-Perez as a weakness of the method used in his LAMA system.

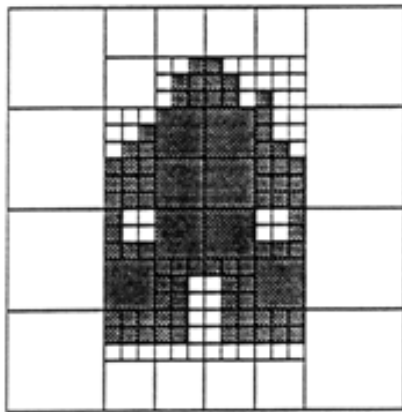
The main advantage of space filling representations has been the simplicity with which they can be constructed, with the simplest case (pixel-size regions) requiring no processing at all to derive the shape representation from image data. For this reason, they are still used in industrial tasks



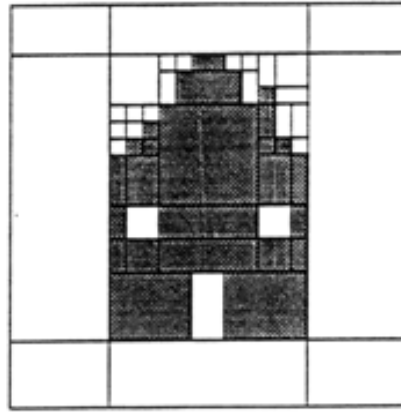
Actual Image



Space Filling from Pixels



Quadtree-like Space Filling



Edge-Relative Space Filling (like LAMA)

Figure 2.1: Space Filling Representations

involving simple shape matching, or location of a single workpiece within the robot workspace. (An example is the detection of a workpiece randomly located on a conveyor belt, which must be visually located so that it can be picked up by a robot arm). Where more complex reasoning is to be carried out, space filling representations soon become impractical.

A secondary advantage of space filling representations arises from the fact that empty space is explicitly represented. This is useful in path planning problems, where empty space of sufficient size for the robot arm to move through can be found directly by searching the scene representation.

2.2.3 Shape Boundary Representation

The most important part of a space filling representation is the way in which the abutment of occupied space regions against empty regions is described. The refinements of space filling methods in the previous section all involve ways of economically improving accuracy along this boundary. A common technique in shape representation for robots is therefore to describe this boundary explicitly, rather than as an implicit line between occupied and unoccupied regions.

Explicit boundary representation provides important benefits both in pure vision systems, and in systems where a robot arm is controlled using a visually acquired shape representation. An example of the former, developed for industrial applications, is quality control and sorting of objects lying flat on a conveyor belt, where boundary shape is the criterion used for sorting. Typical examples are quality control which checks that biscuits are round (no pieces broken off), or a sorting machine for fancy chocolates [Cro82], which stores mathematical models of the curved boundaries of each type of chocolate.

The boundaries of a workpiece are very important in controlling a robot arm, since it interacts with the workpiece only along those boundaries. A good example of this is Trevylan's robot sheep shearer [TKO82], which constructs a mathematical model of the boundary of each sheep. This model is used to guide the shearing arm over the sheep's skin.

Both recognition and robot handling using a simple boundary description are described as goals of a system which, among other tasks, identifies and handles kiwifruit [FA86]. This system describes the "silhouette polygon" of the kiwifruit in terms of a series of vectors around the boundary.

In each of the above cases, the shape boundary is the only information which is relevant to the robot task, so the best shape representation is to

make explicit those parts of the boundary which the robot must operate with.

2.2.4 Constructive Solid Geometry Representation

Boundary shape representation is useful both for visual shape recognition, and for planning robot actions – the first because many industrial recognition tasks can proceed on the basis of boundary information only, and the second because robots only interact with shapes on their boundaries. Boundary representation does have disadvantages, however, where humans must operate with the representation, since we more often consider three dimensional objects to be collections of solid masses, rather than collections of surfaces.

For this reason, robot programming research has resulted in different types of shape representation from those used in object recognition or robot control. Constructive solid geometry (CSG) is often used for this purpose. In a CSG representation, overall three dimensional shape is described as a combination of simpler solids, which in turn can be decomposed, until an atomic level of “shape primitives” is reached.

An early paper by Grossman [Gro76] described the use of procedure calls to represent nested arrangements of parts and subparts in an assembly. The procedural structure of these reflected the hierachically decomposed structure of the assembly.

The approach became more sophisticated with the addition of a “semantic overlay” in a joint representation development project involving Grossman and Lozano-Perez, together with Wesley and Lieberman, the chief architects of the experimental AUTOPASS language [WLPL⁺80]. The “Geometric Design Processor” merges entities from the classes of object, hole, fastener, assembly, and so on, to form descriptions of assemblies in the world. The “world-graph” includes pointers to rigid or non-rigid objects, which in turn refer to their neighbours with friction and constraint relationships.

Generic objects, or particular objects in a scene, can be composed from combinations of solid primitives and “hole” primitives (which describe empty space). The primitive shapes available are cuboid, cylinder, cone, wedge, hemisphere, laminum (flat things), and revolute (solids of revolution). In addition to the shape description facilities of the geometric design processor, the AUTOPASS language provided facilities to describe robot operations at a “human” level, with basic operators such as “INSERT” and “ATTACH”.

The designers of this representation system anticipated that it would be

applied to computer vision, path planning, high level robot programming, and graphical design. In fact, the complexity of the system when compared to methods such as space filling or boundary representation has meant that few attempts have been made to use it so far. The AUTOPASS language has not been completed, so it is difficult to assess the overall utility of the scheme, but it includes many of the concepts that appear in other advanced representations.

Constructive solid geometry representations are discussed further under the heading of computer aided design below, since this is where they have most widely been used.

2.2.5 Object Relative Position Representation

Primitive robot programming systems describe all robot actions relative to the robot itself. Many robot tasks, however, can be described solely in terms of the desired effects upon workpieces. For this reason, “goal-level” programming systems, in which robot actions are not considered by the programmer, describe position of shape elements relative to other objects.

The goal-level robot programming language RAPT allows operators to describe assemblies, with the objective that the description of the assembly should be “interpreted” by a robot. This interpretation would result in the robot creating a physical instance of the described assembly. One of the main goals of the RAPT project was that assemblies be described in terms that are natural to people. With this in mind, the relationships between components of an assembly are defined using operators such as “AGAINST” and “FITS”.

The procedure followed by RAPT involves translating the programmer’s specification of object relationships into geometric descriptions. The relationships between objects are specified in terms of individual shape features of the objects, and the goal state is described as a mapping from the axes of one feature to the axes of another. The features in turn are described with respect to the object body with a mapping to the axes of the object. These mappings are all represented as combinations of translation and rotation matrix operators.¹

¹Further discussion of RAPT can be found in the following references: The RAPT representation of object relationships is described in [AP75], the relational language which is used to implement the system is described in [Pop79], the interpreter is described in [PAB80], and a comparison of RAPT with programming in VAL and with a computer aided drafting system, is given in [APK82]. An extension to RAPT which deals with

2.2.6 Representation for Robot Vision

The greatest variety of shape representation techniques appear to be found in vision research literature. This is largely because the task is simpler than the complete problem of robot control, so systems can successfully employ a greater variety of techniques, without some of the constraints that are imposed by the need to control a physical robot. Greater variety also results from attempts to describe natural objects in outdoor scenes, which can be far more difficult to encompass with formal description techniques than the man-made objects that robots normally act upon.

Many visual representations are optimised for particular visual input facilities, or for recognising particular classes of objects. There is therefore much variation, even between systems that use the same overall approach. Rather than attempting a complete survey, the remainder of this section concentrates on those methods which are applicable to mechanical domains.

The earliest vision systems represented objects simply as a pixel map of the image corresponding to the object. A slightly more sophisticated representation can provide a three dimensional description of an object using “silhouette” bitmaps observed along three axes. The result describes a three dimensional enclosure for the object (excluding closed concavities), which can be stored efficiently. The technique is called rectangular parallelepiped coding [KA86].

The use of edge detection filter algorithms on visual data enabled objects to be represented as a collection of boundaries. Guzman identified different types of vertex that can be formed at edge junctions in three dimensional polyhedral objects (with Huffman and Clowes later providing a theoretical foundation for the classification [RJ88]), and this classification can be used to derive a three dimensional description of an object from the relationships between visible edges.

Lowe has developed a technique for identifying objects in terms of invariant groupings of edges which would have a known appearance when viewed from any angle [Low87]. Shape representation in terms of possible edge groupings only identifies features for shape recognition – it does not provide a full description of the shape. It is interesting in that it provides a consistent mapping from two dimensional to three dimensional representations.

A useful technique for describing three dimensional shape is the method of generalised cylinders, developed by Binford. A generalised cylinder is

representation of tolerance information, and propagation of positional uncertainty through an assembly is described by Fleming [Fle85b].

created by sweeping a two dimensional shape along an axis, with size of the swept cross-section varying according to a sweeping function. A plain cylinder, for example, is simply defined as a circle swept over a straight line, with a constant sweeping function. More complex definitions are easily achieved – a pyramid is a square swept over a straight axis with a linear sweeping function, decreasing toward the apex. Generalised cylinders were proposed as the output formalism for a large MIT vision project by Brady [Bra85a], following Marr’s use of generalised cylinders as a high-level representation.

Description of shape by analysis of boundary features is provided by Brooks’ “Smoothed Local Symmetries” representation, which separates the boundary of a two dimensional image into sub-parts, according to transitions on the boundary that look like joins between parts. Shapes described with this representation are used as the basis for shape generalisation from visual data by Connell [CB87] [Con85], and for a project by Brady in which function of mechanical tools is deduced from their shape [BA84a].

There are many shape representation techniques used in vision systems which can be applied to a more general range of shape than those discussed above. Other methods for representation of shape within a visual image include the use of surface patches of known curvature on three dimensional objects [FH86], the use of “Gaussian curvature” to represent bumps in a more general solid representation than the generalised cylinder [Bli87], and representation of a wide range of natural forms using three dimensional surfaces defined by superquadrics and fractals [Pen86a]. These techniques provide more generality, but are more complex than the above methods, which adequately describe mechanical objects.

2.2.7 Other Robot Representations

Sedas and Talukdar have developed an algorithm for planning disassembly of objects in two dimensions which makes use of multiple representations of the object to be disassembled [ST87]. The three representations used include a sectional view of the assembly, stored as the co-ordinates of all vertices in the section, a connection graph describing points of attachment between parts in the assembly, and a “skeleton” diagram which represents each part as a connected group of convex polygons, with connections between the centroids of the polygons.

Ballard describes a method for representing robot actions using “task frames” [Bal84]. A task frame is a special co-ordinate system based on the object being manipulated by the robot. The task frame moves with

the object, and can thus be used to simply describe tasks which occur in a context that might otherwise add to the complexity of describing robot motion. One such situation would be the assembly of a workpiece while it is moving past on a conveyor. The advantages of the task frame approach are its simplicity of implementation, and the straightforward transformation that can be performed between the task frame and world co-ordinates.

Although the majority of high-level shape representation techniques are designed to be used with visual sensory information, representations can also be built from other sensory input. For example, Briot et. al. outline an object representation which describes an object in terms of the joint positions of a four fingered multiply-jointed manipulator grasping it [BRS78]. This allows objects to be recognised from grasp alone, without any visual or other sensory data. The joint-space object representation is not intended as a general purpose shape description method, but is simply used for object recognition.

These last examples are only a few from a great range of representation techniques, but they indicate some of the variation that is possible in describing shape and space for robot reasoning. The previous sections have surveyed the most influential techniques in spatial representation for robots, but it has been necessary to omit a number of interesting systems.

2.3 Shape Representation in Other Fields

Most general purpose approaches to shape representation have been developed outside the field of robotics. The study of “Solid Modelling” has application not only to robotics, but to computer aided design and drafting, and also computer graphics.

Solid modelling techniques have developed with growing sophistication in CAD methods, which have progressed from a simple description of lines and points in an image, to description of three dimensional objects in terms of “wireframes” (the edges and vertices of the three dimensional shape), to boundary representation, which describes the surfaces of an object, and constructive solid geometry, which describes solid shapes in terms of primitive solid components. The remainder of this section discusses each of these categories of representation technique.

Computer aided drafting was first done by specifying a set of co-ordinates of lines and points in a format which could be stored, edited, and used to make copies of the defined drawing. Specifications were either entered

numerically, or captured from x-y digitising devices. CAD systems today have generally evolved from this level, but many are still related to manual drafting only in the same way that a word processor is related to a pencil – they remove the tedium involved in copying and editing, and sometimes make data entry more efficient. They do not, however, have any explicit representation of the object being drawn; The internal representation of the drawing is simply a collection of the lines and points which will appear on the page.

2.3.1 Wireframes

Solid modelling systems are distinguished from simple CAD in that they allow three dimensional objects to be designed in a three dimensional space. The three dimensional representation can then be used to automatically generate two dimensional projections.

The most primitive form of CAD solid modelling is the “wireframe” method. A wireframe is a description of the vertices and edges of a three dimensional object, specified by x, y and z coordinates. The wireframe description of an object can be used to generate computer drawings in various projections, and can be manipulated by an operator for such functions as on-screen rotation. Wireframe systems do not normally perform “hidden line removal” and the operator must therefore specify the visibility of each edge and vertex for a given projection.

Wireframes are not sufficiently powerful for many spatial reasoning tasks, because it is possible for a single wireframe to represent several different objects, depending on what spaces in the frame are filled – wireframes are ambiguous. They are not altogether satisfactory for CAD tasks either, because they allow the accidental creation of “impossible objects”, such as a cube with one edge missing.

2.3.2 Boundary Representation

An obvious extension to the wireframe method was to represent the polygons composing an object surface, rather than the edges only. This type of method is known as “boundary representation”, because it describes the boundaries of the solid (in terms of an enclosing surface). Extensions to polyhedral object representations include the use of surfaces specified from b-spline curves. B-spline boundary representation techniques are used heavily in current computer graphics technology for visual effects involving three

dimensional objects, and are also sufficient for some machine tool control applications, where the computer system must know the goal shape of the workpiece surface after cutting.

Boundary representation methods are more powerful than wireframes, because they represent an object in terms of its surfaces, rather than its edges. A wireframe can thus be created from any boundary representation, but not necessarily vice-versa (because of possible ambiguity). The use of surfaces prevents the “missing edge” problem, and can also prevent ambiguity. Some checking is still necessary, however, to ensure that a given collection of surfaces describes a valid three dimensional object.

2.3.3 Constructive Solid Geometry

Constructive solid geometry systems allow the definition of complex three dimensional objects using a combination of simpler objects. A CSG system normally includes a set of solid shape primitives such as cubes, cylinders, cones, etc. which can be defined in a range of sizes and shapes. These primitives are combined using “set operations”, or “boolean operations”, which allow the shape of an object to be described as the intersection of other objects, or as the union of objects. Inverse sets (or subtractions) are used to make holes in the overall shape. The resulting shapes can in turn be joined, or intersected with other shapes, to describe three dimensional shape of any complexity. CSG systems often include facilities for defining new primitives by “sweeping” operations, such as those that are used in the generalised cylinder method described in the previous section. Two well known CAD research systems that use CSG methods are GMSolid [BG82], and PADL-2 [Bro82b].

Constructive solid geometry systems are more powerful than boundary representations in the same ways that boundary representations are more powerful than wireframes. A boundary representation can be automatically derived from a CSG description (in fact this is often done for graphical display purposes). Objects constructed using CSG are guaranteed to be valid three dimensional objects, and such a representation cannot be ambiguous. CSG representations have been extended in various ways; examples are the inclusion of tolerance or uncertainty data [RC86], or object-oriented style “methods” knowledge for operating on defined objects [Big86] [Ela86].

Recently developed systems have included combinations of the above methods. Gossard, Zuffante, and Sakurai describe shape using a combination of CSG and boundary representations [GZS88]. Their “Object Graph”

can include both solid primitives and boundary surfaces in the description of an object.

2.3.4 Other Methods

Both constructive solid geometry and boundary representation techniques are also used outside of the engineering domain, most notably in computer graphics work. The most difficult problems in computer graphics seem to involve the realistic representation of natural objects, rather than man-made ones. This means that new techniques developed for graphics systems, such as texture mapping, have not yet influenced the fields of either CAD or robotics, which deal mainly with man-made objects.

A number of fields make extensive use of computerised shape representation, but are not concerned at all with solid shape, because they are building on commonly used two dimensional abstractions. A typical example is computer assisted mapping, where there is a definite correspondance between the two dimensional map and the three dimensional shape of the land, but mapping conventions allow all infomation about the map to be described in two dimensions only. Such systems may include sophisticated two dimensional shape representation (such as extended polygons, in Geovision's "AMS" system), but represent three dimensional shape only in that they allow the draughtsperson to draw contour lines.

Another example of a two dimensional abstraction that has lent itself to computerisation is plane geometry. Gelernter's 1963 geometry-theorem proving machine used descriptions of two dimensional shape that specified a range of possible coordinate values for each point in the shape described, thus allowing the system to test its proofs over a range of cases.

This chapter has covered a variety of methods for representation of, and reasoning about, shape and space in two and three dimensions. Several of the methods discussed here have influenced the development of the qualitative shape representation presented in chapter 4, and they will be referred to again in that chapter. This survey has also given some indication of the limits of spatial reasoning capabilities both in robotics and in other fields – limitations arising from lack of accuracy, ambiguity, or difficulty in constructing the representation either from a vision system, or from a human programmer. Chapter 6 discusses ways in which a qualitative spatial representation can overcome some of these limitations.

Chapter 3

Qualitative Reasoning in Spatial Domains

3.1 The Origins of Qualitative Reasoning

There have been two main streams contributing to the development of the field of “qualitative reasoning”. The first of these, usually referred to as qualitative physics, builds on the work of de Kleer, who pioneered the use of qualitative methods in solving engineering problems. The second is a program initiated by Hayes, with the aim of developing a “Naive Physics”, which Hayes defines as “a large-scale formalism” of commonsense knowledge.

Nearly all published work in qualitative reasoning acknowledges both de Kleer and Hayes, but the majority follows the example of de Kleer’s work, in that it attempts to analyse specific physical situations in qualitative terms. In contrast to this approach, Hayes recommends that naive physics should consider a wide range of human experience in developing qualitative models. The following two sections present firstly a historical overview of developments in qualitative physics, and secondly a brief discussion of the aims of naive physics, which considers how far qualitative reasoning research has progressed toward those aims.

3.1.1 Early Qualitative Reasoning Systems

The first (1975) published paper which is referred to in qualitative physics literature was de Kleer [dK79]¹. The system described in this paper was intended to model human problem solving in Newtonian mechanics. Several people had previously worked on systems which could solve problems at the level of “high school physics”, but de Kleer was the first to use qualitative methods in an attempt to model the thought process of a student more closely – the others simply extracted numbers from a problem description, and used linguistic cues to select the mathematical operations which might be appropriate.

De Kleer’s system, called NEWTON, described motion of a block on a roller coaster in terms of the speed of the block and the slope of the roller coaster. The speed could have one of three qualitative values – positive, negative, or zero. The slope of the roller coaster was likewise described by one of three qualitative slopes – up, down, or level. NEWTON performed an initial qualitative analysis to find places at which the behaviour of the block was likely to be interesting, and later applied numerical techniques if operations using the qualitative values were not sufficient to reach a conclusion about behaviour at particular locations. The use of multiple representations to support both qualitative and numeric reasoning was the major emphasis of the research.

In 1979 de Kleer extended qualitative techniques to the domain of electronic circuits². The techniques used for reasoning about circuits retained NEWTON’s use of three qualitative values. An emphasis of the work was establishing causal relationships between values at different nodes of the circuit. The use of the circuit domain removed any need for spatial representation in this research, since a circuit can be completely represented by its topology, without any spatial information.

In 1980, de Kleer and Brown described a system which modelled human understanding of mechanical devices [dKB80], [dKB83]. The input to the system was a structural description of the device. From this information the system would create an “envisionment” of its likely behaviour. The example used was an electro-mechanical buzzer. The analysis of the buzzer

¹The reference listed in my bibliography is a version of this paper in a 1979 collection. The paper was initially published as MIT AI Lab Technical Report TR-352 in 1975. De Kleer also published related papers in 1977 [dK77]

²Initially described in “Causal and Teleological Reasoning in Circuit Recognition”, MIT AI Lab Technical Report TR-529 1979, but more fully developed in [dK84]

proceeded by analysing how some components could cause changes in the state of others, thus changing the state of the whole system. The oscillation of the buzzer could be deduced from the closed loop in the state transition diagram.

In 1981, Forbus published a description of a system which reasoned about the motion of a bouncing ball through a two dimensional scene consisting of free space bounded by vertical and horizontal surfaces [For81]. “FROB” could, given a description of the initial trajectory of the bouncing ball, predict where it might come to rest. The main tools used by the system were the “metric diagram” and the “place vocabulary”. The “metric diagram” was a representation of the scene that was intended to model the human capability to reason using an internal image of a situation [For83]. The “place vocabulary” was a subdivision of free space into regions with different qualitative features. Motion was then described as a transition between regions. This part was purely qualitative, and included no numerical analysis components.

Forbus developed a new technique for qualitative reasoning in 1981, which he called “qualitative process theory” [For84a]. Qualitative process theory describes physical quantities such as temperature and pressure in terms of a “quantity space”, which is an ordering of values that have qualitative significance for the process being described (for example – boiling point is a qualitatively significant value for temperature when a water heating process is being described). The process is described as a set of “influences” on quantities associated with objects in the situation described. For example, heating might be described an influence exerted by a heater on the temperature quantity associated with a fluid.

A large body of research has developed from Forbus’ qualitative process theory, and from de Kleer’s work on causal reasoning about circuits. A comprehensive collection of this work is the book “Qualitative Reasoning about Physical Systems”, published in 1984 [Bob84a]. Since this time, the field has continued to grow, and I will mention later work only if it has a bearing on spatial or mechanical reasoning.

3.1.2 Naive Physics

“Naive Physics” is a term coined by Hayes [Hay78] in 1978, to describe his approach to developing a “large-scale formalism” of commonsense knowledge about the world. This concern with real world knowledge can be related to a general awareness amongst AI workers that future progress in AI depends

on intensive knowledge being made available to reasoning systems.³

The aim of naive physics as stated in [Hay83] is to formally describe the world in the way that most people think about it, rather than describing it in the way that physicists think about it. This description should attempt reasonable completeness – that is, it should describe a significant portion of the way we understand the world, rather than just small pieces.

The use of the word “naive” indicates that this description must include commonsense knowledge that is normally taken for granted in formal physics, and it therefore may include elements outside what we consider to be the field of physics. It may also choose to describe phenomena in a way that is familiar to “the man in the street”, but would not be considered appropriate to a physicist. Two examples are the “force” of sucking, and “impetus” theories of motion, both of which adequately describe everyday phenomena, and are prevalent theories amongst intelligent people [McC83], even though they are considered to be inappropriate for physicists.

The “Naive Physics Manifesto”, and the its revision in the “Second Naive Physics Manifesto”, proposed the construction of a formalisation of commonsense knowledge which covered a broad range of knowledge using a common framework, and included dense factual detail. It specifically did not recommend the construction of programs or new formal description methods. It did dwell on the importance of spatial representation, although it excluded the sort of spatial reasoning that is necessary to plan physical motion⁴.

Nearly all qualitative physics literature has cited the “Naive Physics Manifesto” (including Forbus [For81], Faltings [Fal87], Kuipers [Kui82], Stanfill [Sta83a], etc.), and it can therefore be considered to be a foundational work for qualitative physics.

The wide influence of the manifesto can be easily accounted for – it is compelling reading, pointing out clearly the deficiencies of much work in A.I., while proposing a clear, exciting, and apparently practical vision for future progress. However, most qualitative physics work does not follow Hayes’ recommendations. In particular, an emphasis on restricted domains, and on construction of reasoning programs rather than on knowledge representation, separates qualitative physics from naive physics as proposed by Hayes.

In addition, the importance of spatial reasoning in the commonsense

³Hayes’ awareness of the problems of real world reasoning may also have been strengthened by his early work in robotics.

⁴[Hay83] p.10

world as pointed out by Hayes has not resulted in an emphasis on spatial problem solving in qualitative reasoning. On the contrary, most work concentrates on problem domains which can be abstractly represented without spatial information. This thesis addresses some issues in spatial representation which are obviously related to commonsense spatial reasoning ability, and they may assist progress toward some of the goals that Hayes has set.

3.2 Techniques in Qualitative Reasoning

This section presents an overview of the reasoning techniques which have been employed in qualitative reasoning systems. There are a few central methods which are employed in almost all qualitative reasoning systems, while others are applicable only to certain types of problem. I have distinguished between these cases, and also note which methods are particularly relevant to spatial reasoning.

3.2.1 Classification of Qualitative Reasoning Tasks

The problems which have been used as test cases for qualitative reasoning systems can be classified in several ways. They can be classified by the general domain in which they occur (medicine or engineering, for example), by complexity, by the quantity of information involved, and so on. They can also be classified by the problem formulation, and the general reasoning strategy that must be taken to find the problem solution. The following list describes types of problem formulation that require different reasoning strategies.

- Explanatory tasks start from the structure of a system, and produce an explanation of behaviour in terms of that structure. Behaviour includes the response of the system when it is perturbed in a given way. De Kleer and Brown's work has emphasised this type of task; the hydraulic pressure regulator circuit is an example of deriving function from structure.
- Predictive tasks start from a description of a system which is in an unstable state. From the structure of the system, and the initial state, they produce a prediction of either a sequence of future states, or a final stable state. De Kleer's roller coaster problem, and Forbus' bouncing ball are both example of predictive tasks.

- Diagnostic tasks start from a state or behaviour that is inconsistent with the presumed structure of the system. The system specifies what element of the structure could, in changing, produce the observed behaviour. A typical domain for diagnostic tasks is electronics, where the system must identify a faulty component. Davis [Dav84b] describes one such system.
- Planning tasks start from a description of a system that is in a given state, together with a goal state for the system. The system must describe a perturbation of the system which will cause it to reach the goal state.
- Design tasks start from a description of desired behaviour for a system, and produce the structure of a system which will have the desired behaviour.

Planning and design tasks are particularly appropriate for the application of qualitative techniques, but they are not usually attempted, perhaps because they are less constrained than the other three types of task. The problem solving system described later in this thesis does carry out a planning task.

A further classification of qualitative reasoning methods can be based on whether the problem domains in which they operate are discrete or continuous. A typical discrete system can be described as a set of separate components, each of which has one or more “ports” by which it can be connected to other components. The state of a discrete system is the sum of the states of all components in the system at a given time, where a component state is described in terms of the values at each of its ports. Typical discrete problem domains are those of electronic circuits, and fluid circuits, where state is expressed in terms of voltage and current, or pressure and flow, at each port of a device. The connections between ports are regarded as “conduits”, which guarantee equal values on each side of the connection [dKB84].

A continuous domain, unlike these discrete domains, cannot be completely described by topology. The “conduit” in discrete domains is actually a descriptive device which explicitly discounts the spatial aspects of connections between nodes, allowing physical devices to be represented as abstract networks. Using these techniques for spatial reasoning problems would result in a loss of much spatial information. The normal approach to qualitative reasoning in continuous domains however, involves exactly this

type of method. The domain is made to look more discrete, by dividing the continuous space into qualitatively distinct regions. Motion can then be described as a series of transitions between discrete states – each region is represented as a possible state.

The roller coaster problem is an example of this approach, where a linear space (the length of the roller coaster) is divided into distinct regions according to slope. An object travelling the roller coaster can, from any state, make a transition into one of two other states, giving it one degree of freedom. The bouncing ball is an example where there are two dimensions in which qualitative division takes place, and hence two degrees of freedom for state transitions (up/down and left/right).

The significance of this “discretising” of continuous spatial domains will be considered further in later sections, but I will first describe the common elements of qualitative reasoning programs that operate on the function and structure of topologically described discrete systems.

3.2.2 Elements of Qualitative Reasoning

The most fundamental common feature of qualitative reasoning systems is the use of the “quantity space”. This term derives from Hayes’ “quality space”, which in naive physics refers to a set of possible values for properties of an entity, where the properties are meaningful “independently of the entities which possess them.” Distance, therefore, is a quality space in naive physics, because distance exists independently of particular entities.⁵

Forbus uses the term quantity space to describe an ordered set of qualitatively differentiated values of a particular quantity. The values that are present in the set are not universal, but apply to the current problem. The boiling point of water, for instance, might appear in the temperature quantity space of some problems, but not in the quantity space of a problem involving oil. The meaning of the term quantity space has been extended to apply to any set resulting from a qualitative division of a continuous value range. It has been used, for example, to describe the three-valued system used by de Kleer in the original roller coaster problem (and by many others since) – positive, negative, and zero.

A number of alternatives to the quantity space have been proposed, using

⁵Hayes also uses the term “quantity space”, but by it he means any space in which reasoning takes place without knowledge concerning the properties of the space. This difference in terminology can be a source of confusion when investigating the relationship between qualitative physics and naive physics.

various methods for dividing a continuous value range into discrete qualitatively distinguished regions. These methods include the “fuzzy logic” of Zadeh [Zad79], which has been used for qualitative reasoning by D’Ambrosio [D’A87], Simmons’ “Commonsense Arithmetic” [Sim86], and Raiman’s “Order of Magnitude” reasoning [Rai86]. The most essential feature for qualitative reasoning, whichever of these techniques are used, is that the “discretisation” is done with reference to landmark values in the problem itself, rather than with respect to arbitrary ranges.

Another fundamental concept in qualitative reasoning is “envisionment”, which refers to the process of predicting and analysing changes of qualitative state. Important programs which perform envisionment on discrete systems are ENVISION, by de Kleer and Brown [dKB82] and QSIM, by Kuipers [Kui82]. The transitions between states are determined by causal relationships between components of the system.

In discrete systems, the future state of the system can be analysed in terms of individual component behaviour. Envisionment involves predicting the series of qualitative states that will result from any perturbation to the system. The envisionment depends on system topology, and on component properties, because the perturbation propagates between discrete components of the system. The results of envisionment can be used to make conclusions about function, and to determine stable states (in which any small perturbation will tend to return the system to its current state).

As well as detecting stable states, the envisionment method can predict instability. This has led to a number of explanatory programs that use oscillators as test examples – both mechanical, electrical, and pneumatic oscillators (see Falkenhainer et. al. [FFG86]). It has also encouraged the use of qualitative reasoning methods to analyse feedback systems, since the stability of a feedback system is one of its most important properties.

The envisionment technique can be extended either by the use of “independent experts” (as in de Kleer’s roller coaster, where a mathematical analysis could be used to define more precisely an envisioned qualitative state), or by more sophisticated envisionment (as in Kuipers’ QSIM algorithm, or Forbus’ qualitative process theory, which make use of derivative information as part of the system state in addition to magnitude of quantities).

Recent developments in qualitative reasoning applied to discrete domains have largely retained the de Kleer and Brown envisionment structure, and have concentrated on more sophisticated representation of causality through state transitions, or on more sophisticated organisation of the

quantity space. (See [Wil86], [IS86a], [dKB86], and [IS86b], for example). On the other hand, a number of qualitative reasoning workers have become aware that these programs share a lack of power in spatial reasoning, and they have begun concentrating on applying qualitative techniques to spatially complex domains that cannot be adequately represented as networks of discrete components.

3.3 Spatial Qualitative Reasoning

The first of the qualitative reasoning systems described above was de Kleer's Newtonian physics problem solver, which operated on the roller coaster example. This example is set in the spatial domain, in that it involves continuous length and height representation of the roller coaster. However, as mentioned, it has only one degree of freedom, because the height component of the motion is dependent on the length component.

This allowed the representation to be simply an ordering of qualitatively different segments of the roller coaster, each with an associated height and slope (the system also required a complete geometric description of the roller coaster, but this information was not used by the qualitative reasoning component). A similar one dimensional system was designed by Forbus, which analysed the one dimensional motion of a block on a spring [For82].

These systems, although operating on problems that involve spatial motion, allow the problem to be abstracted to a degree that it can be easily analysed using techniques for systems without spatial components. Many of the problems that are involved in real world motion analysis are not considered in these systems – the most important of which are the representation of direction of motion, and of relationships between objects.

3.3.1 The Bouncing Ball World

The bouncing ball problem solved by Forbus' "FROB" involves more complex two dimensional reasoning, because there are two degrees of freedom in the ball's motion. FROB represented the problem in two different forms: the "metric diagram" was a geometric description of the scene, which was essentially the formatted input to the qualitative reasoning program. From the metric diagram, the system computed the "place vocabulary", which is the set of possible states for the moving ball, together with information about possible transitions between those states.

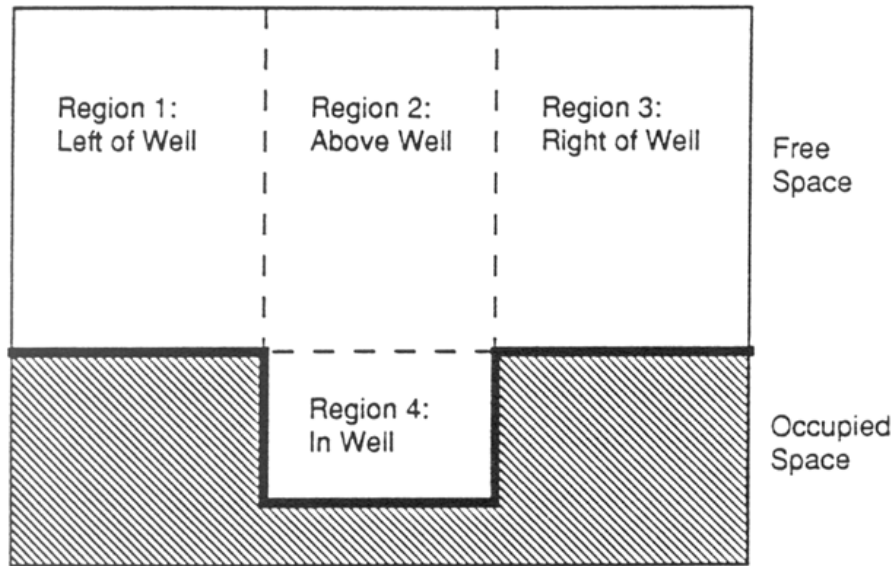


Figure 3.1: Free Space Division Guided by Scene Features

FROB divided the free space in the metric diagram into qualitatively distinct regions by extending discontinuities in the free space boundaries. This means that the presence of a “well”, as shown in Figure 3.1, results in the division of the otherwise uniform free space into four regions – in the well, above the well, to the left of the well, and to the right of the well.

The spatial regions in the divided metric diagram correspond directly to possible qualitative states (actually temporal locations bounded by time interval and spatial position) in the motion of the ball. From any one region, there are a finite number of other regions that the ball can move into (at most eight, including diagonals); these possible motions are represented as possible state transitions. Forbus calls this representation the “place vocabulary”.

The division into regions from the metric diagram is always carried out along vertical and horizontal lines constructed through boundary discontinuities. The vertical direction is defined with reference to gravity, and the horizontal is perpendicular to gravitational acceleration. The effect of this is to include implicit information about gravitational acceleration and potential energy in the scene representation – speed of motion is constant in the horizontal direction, and acceleration is constant in the vertical.

The current state of the moving ball was described in terms of location (a “place” in the place vocabulary), and direction of motion. The next state could be determined from this information, since the place vocabulary included information about how individual places are connected to other places in each direction. The program could then derive an envisionment of possible future states, given the current state, together with the place vocabulary.

A later paper by Forbus [For82] discusses the application of his qualitative process theory to the bouncing ball problem, with motion being described as an “influence” on the “position quantity” associated with an object. Acceleration and energy are also treated as quantities which are influenced by processes. The implemented example which he describes involves one dimensional motion of a block and spring – a classic example of energy conversion. He does not suggest a definite approach for extending the energy conversion of this example to the two dimensional case of the bouncing ball, but in a more recent paper gives the opinion that “quantity spaces don’t work in more than one dimension” [FNF87].

The approach to qualitative two dimensional reasoning which I have developed can use a quantity space technique to describe two dimensional motion and position – in terms of proximity relationships between objects. This technique is described in chapter 4.

3.3.2 The Mechanism World

Just as many discrete envisionment programs operate in the domain of circuits (either electronic or hydraulic), many qualitative spatial reasoning programs operate in the “Mechanism World”, or “Machine World”. This domain involves analysing motion in systems of rigid mechanical components. The interaction between components includes relative motion, mutual constraint, and transmission of kinetic energy.

The reasons that this domain is popular appear to be firstly that it has been thoroughly analysed at a theoretical level (in mechanical engineering), and secondly that behaviour of mechanical devices is usually simple and consistent – they have been designed with the objective of achieving predictable and repeatable motion.

Two systems for the qualitative analysis of mechanisms are those produced by Stanfill [Sta83a], and Joskowicz [Jos87]. Stanfill’s “Mack” system accepts a geometric representation of a simple machine, and applies rules in solid geometry, mechanics, and pneumatics, to produce a qualitative descrip-

tion of the machine's behaviour. Typical machines analysed by Mack are pistons and cylindrical bearings. Mack operates by building successive models of the system in terms of pneumatics, forces, acceleration, and process, with the final process model being an expression in the form of Forbus' qualitative process theory; it relates motions to pressures and to other motions. The spatial analysis component of this system uses geometric techniques, and it is only the more abstract levels that are expressed in qualitative form.

Joskowicz describes a system which reasons about kinematic chains. A kinematic chain is a series of kinematic pairs. A kinematic pair is a mechanical device which transmits kinetic energy, such as a pair of cogs, or a belt and pulley. Each pair involves two components in contact, with at least one axis of possible relative motion (i.e. they are not rigidly joined). Kinematic pairs can enable the transmission of kinetic energy between components, and they can also provide various types of constraint on the motion of components.

This system initially analyses the possible axes of motion for each pair of components in the chain, using a configuration space derived from a constructive solid geometry description of the components. A qualitative environment is then carried out to determine the operation of the whole chain, with kinetic energy being treated as a material that is affected by each kinematic pair. This stage of the analysis is closely related to the techniques used for circuit or fluid flow analysis, and described above. The spatial analysis, however, uses standard geometric techniques for determining configuration space from a CSG description, rather than qualitative methods.

Neither of these two systems operate from first principles in analysing relative motion of a pair of components in contact; instead, they classify each combination of components into a known category. Joskowicz's system uses Reuleaux's 1876 classification of kinematic pairs, for instance.⁶ Faltings considers that this approach lacks power because the system cannot deal with situations that are not represented in its symbol set [Fal86] [Fal87].

Forbus, Faltings and Neilsen describe the application of Forbus' metric diagram and place vocabulary (or MD/PV) approach to the "mechanism world" [FNF87]. State is represented in terms of connectivity, or types of contact between objects. The metric diagram provides quantitative information from which one can calculate a configuration space for the component, given constraints on its motion. This configuration space is then subdivi-

⁶Franz Reuleaux "The Kinematics of Machinery: Outline of a Theory of Machines" Reprinted by Dover Publications in 1963

vided according to contacts that can occur between components. The place vocabulary is created from this information, and qualitative motion can be envisioned in terms of changes in state between these contact configurations. For example, the turning of a ratchet involves state transitions from contact between one gear tooth and the pawl, to a later state of contact with another gear tooth.

The MD/PV model also aims to solve problems involving kinematic chains, and Faltings' goal as described in [Fal86] is to explain the action of a clock after an analysis of individual kinematic pairs within it. An envisionment of motion in a complete mechanism such as this would involve operating on very complex system states that include connectivity information for all components of the mechanism.

Plimmer's study of kinematic chains as found in the analysis of a bicycle [Pli85] makes use of a simple three-valued quantity space in determining direction of motion for bicycle components such as cogs and chains. Causal motion relationships between components are represented explicitly by interaction slots in a functional representation of each component in the kinematic chain. The bicycle analysis system does not include any notion of state, and deals only with a bicycle which is in a "static" condition (actually in constant motion, without acceleration or deceleration).

Systems that operate in the "mechanism world" all reason about contact between components, and contain only components that have motion constrained by axes or joints, so that there are no more than two degrees of freedom in motion (only one – direction of rotation – in Faltings, Stanfill and Plimmer). The system which I describe in chapter 5 can represent objects moving in free space, as well as objects in contact. The above systems use quantitative geometric techniques to establish fundamental motion constraints from shape description, in particular Lozano-Perez's configuration space method (this is described in [LP83]). The system I describe is interesting in that it derives possible motion from shape using qualitative methods only.

3.3.3 Other Spatial Reasoning Systems

Other qualitative spatial reasoning techniques have been proposed that vary widely from the combination of quantity space, system state, and envisionment shared by most of the systems described above.

Examples of these are Schmolze's "Physics for Robots" paper [Shm86], which tries to extend Hayes' ontology of liquids to the situations that would

be encountered by a robot in a kitchen and Davis' "Ontology of Physical Actions" [Dav85], which has similar aims at a more general level – a formal theory of solid objects that humans encounter – for use by a robot. Davis also describes a representation that can be used for reasoning about spatial relationships between objects in terms of containment and relative location [Dav84a].

These examples are normally classified as "Naive Physics", since they attempt to find a formal basis for reasoning about very fundamental human concepts of space, rather than trying to solve any particular problem of motion or shape description. Work of this kind may become important to the type of qualitative reasoning described above if it is to be incorporated in a more general system for reasoning about action in the world. At this stage, work relating to qualitative spatial reasoning can proceed without requiring study of these issues.

Some use has also been made of qualitative methods for solution of real-world problems, because of the advantages of qualitative data as a basis for rule-based decision making. Two examples of this are a system for qualitative description of motion and position in an autonomous vehicle [BB87], and one for qualitative description of plug and socket shape when mechanical parts are to be fitted together [GB87]. These projects give examples of approaches to qualitative representation, but do not use qualitative methods (such as envisionment) for any type of planning or analysis. The representations developed for these systems tend to have less generality, since they are concerned only with distinguishing between specific situations or objects, rather than with general techniques for qualitative spatial reasoning.

The system which I describe in chapters 4 and 5 also departs in some measure from the mainstream of qualitative reasoning, but is more closely related to the mainstream than those of Schmolze, Davis, Burger and Bhanu, or Green. Its representation of state is more general than the "mechanism world" systems, in that a given state describes the complete relative positions of all objects in a scene (not just contacts). It uses a space description method that is related to the quantity space, but this is adapted so that Forbus's reasons for criticism of the quantity space in two dimensional situations are avoided. The main features of the approach, however, were initially developed from consideration of robotic reasoning and solid modelling systems, rather than from other qualitative reasoning projects.

Chapter 4

Two Methods for Qualitative Representation of Shape and Space

This chapter describes the development and implementation of a qualitative geometric representation. The representation facilitates reasoning methods that overcome some of the limitations of AI systems discussed in previous chapters – in particular, the limitations of robots that cannot solve problems using “commonsense” spatial reasoning, and the limitations of experimental qualitative reasoning systems that cannot make use of spatial information.

The qualitative geometric representation incorporates ideas and techniques that have been described in the previous two chapters, and are now used in a new overall context. The first part of this chapter discusses issues that need to be considered while developing any spatial representation or geometric modelling system if it is to be useful in robot reasoning. The second part considers how two major solid modelling techniques can be adapted to qualitative methods, and describes how each resulted in a different approach to two dimensional scene description. The third section presents these two different approaches in detail, discussing the advantages and disadvantages of each, and the fourth section discusses further extensions which have not yet been implemented.

4.1 Representation Issues in Qualitative Robot Reasoning

Comparing the way that humans think about physical situations to the way that robots are currently programmed suggests a number of important issues that should be considered in the development of new robot reasoning systems. In this section, I concentrate on the issues arising from four aspects of human spatial reasoning that appear to be particularly important in solving simple qualitative spatial reasoning problems. These aspects are:

1. *Representation of detail at multiple levels.* People are able to store a large amount of detailed information about a complex object, yet also consider that object in terms of gross shape alone when this is necessary. They are also able to focus on a particular detail of the overall shape, while retaining a record of its context. An example of this ability is the way that a mechanic views a car. He knows a huge amount about its detailed shape, but is able to think, when necessary, simply in terms of its overall shape (when driving it).
2. *Independent reasoning in local contexts.* Where overall shape is very complex, people are able to reason about one part of the overall shape, treating it as an independent context. The car mechanic for example, when fixing a handbrake, is able to work purely with that local context within the car. Other contexts, such as the seats or headlights, can be temporarily forgotten. He can also fix a truck handbrake as easily as a car handbrake, by operating in a local context that is abstracted from the overall shape of either vehicle.
3. *Assignment of properties to groups of features.* People are able to assign an abstract description to a whole set of shape features, and then make statements about the new abstraction, rather than simply about a single instance of it. This ability is particularly noticeable in its absence – sufferers from visual agnosia are unable to construct abstract descriptions of sets of details in their visual field, and are thus unable to recognise generic classes of object, even though they can see all of the details.
4. *Qualitative size description and judgement.* In many spatial reasoning situations, the absolute size of a given shape feature is not important.

Its size relative to other shapes may be more important, as in the question “will this washer fit over that bolt?” Alternatively, its size may be altogether irrelevant, as in the question “is that a bolt?”. If qualitative reasoning methods are available, it is possible to discuss relative size, or size-independent questions, without numerical information.

4.1.1 Representation of Detail

A scene description used by a robot may include much information that is not relevant to the task currently being performed. This is especially likely when the robot has acquired the scene information by sensory means, whether visual, tactile, or in the form of range data. The irrelevant information is usually superficial detail, but even overall shape may be irrelevant to the task.

One of the most important functions of robot sensing systems is the fact that they filter raw sensory data, and provide a description of the object or scene sensed at a higher level of abstraction (for example, they may take an array of pixel brilliance values, and produce from it a description of object edges).

Some form of filtering is always necessary, but the requirements of the resulting filtered description can vary widely. A system which sorts nuts from bolts could abstract its sensory information to the degree that it describes any object as either `nut`, `bolt`, or `not-nut-or-bolt`. A system that joined nuts and bolts together, however, might need to know what head type the bolt had, what diameter it was, and what thread standard it conformed to. A system that manufactured nuts and bolts would need to know the depth of the thread, the thread spacing, the groove angle, and so on. A quality inspection system might need to know the exact position of every point along the thread.

A general purpose reasoning system, if it were to imitate human capabilities, should be able to separate these levels of detail, so that they are all available to it when necessary, but so that suitable abstractions can be used without considering what lies beneath them. This approach can provide more reasoning power than an arbitrary level of abstraction, because it is not always possible for a sensing system to select an abstraction without an understanding of the task – particular shape details may or may not be important in the context of a given task.

As an example, the irregularities in the surface of a rough cast fireplace grate are completely irrelevant to the function of the grate, but the general

roughness of the casting surface may be quite important to a robot which is assembling fireplaces, because the roughness of the grate prevents it from sliding over other surfaces. In this case any given irregularity on the grate is still of no interest to the robot – it is texture that is important.

A contrasting situation is that of a large mechanical part, which is to be oriented in an assembly with the aid of a locating pin. The shape of the part as a whole might be irrelevant to the orientation task, while the shape of the small locating pin is very important. The contrast between this example and that of the rough casting makes it clear that the significance of a given detail in an object's shape depends on the functionality of that shape detail, and on what use a robot is trying make of the object.

A shape description could reflect this dependence on the functionality of shape elements by including in the representation only those elements of the shape which are functionally important. This is the de facto methodology of current robot programming, in which a programmer must decide whether or not a given aspect of the object's shape will ever be available to the robot program. These decisions are encoded in the program when the object description is being constructed by the programmer (I refer here to task-level programming languages such as RAPT, where objects are explicitly described, rather than robot-level languages, in which the shape of the object is only implicit).

A more intelligent robot, acting on the basis of representations that have been acquired from sensory data, will not necessarily have the information available which is needed to make decisions about the function of parts at the time that the representation of an object is constructed. Such a robot cannot ignore small details of object shape, because they may become important during operations on the object. On the other hand, the inclusion of every detail in a complex shape description may hinder the construction of useful abstractions for the robot reasoning system to work with.

As an example of the need to judiciously ignore detail, consider a six-pack of beer bottles. A detailed description of its shape appears to be very complex, and could obscure properties such as the fact that sixpacks can be regarded as rectangular blocks for the purposes of stacking them. A person wondering whether sixpacks could be stacked would not stop to analyse the shape of each bottle top. They would notice the overall box-like shape of each sixpack, and experiment with stacking them on the basis of this coarse description. A robot which is reasoning about its workspace in a qualitative fashion should be able to perform such a task with a similar economy of detailed analysis.

A way of achieving both simple overall shape description and retention of details which may become functionally important is to represent objects at multiple levels of detail. This approach has been used in computer systems that must perform high level analysis from possibly noisy input data, in domains such as reading handwritten script [SB87], or recognising speech [dM87]. The explicit representation of multiple levels in these systems allows them to continue referring to low level sensory information even after high level analysis has commenced.

Most machine vision problems are amenable to exactly this approach – they have a large amount of input data (typically the brightness of every pixel in the vision field), and they filter it to provide a higher level abstraction. A multilevel shape representation might involve making links from high level description to the low-level data. People, however, do not store large amounts of information that is later filtered – they store a coarse description as a “first impression”, and collect more detail if necessary by focussing their attention [Pen86a]. Some attempts have been made to provide machine vision systems with similar controlled focus facilities, as described in [Fun80] [Pen87].

There are two options, therefore, in providing a vision system with the ability to create scene descriptions containing multiple levels of detail. Whichever is used, the reasoning processes described above can be supported – the difference lies in the visual control mechanism, and in the data storage techniques used. This section has argued that it is both useful and plausible for robots to represent shape data at multiple levels of detail. The shape representations discussed later in this chapter support qualitative methods which can make use of this type of sensory data, and perform this type of reasoning.

4.1.2 Geometric Reasoning Using Local Contexts

A major disadvantage of the space filling techniques for object representation discussed in chapter 2 was that the description of an object depends on its orientation with respect to the axes used to partition the space, and its apparent size with respect to the axis units. The axes themselves often have no functional relationship to the task being performed, and the variance in descriptions disguises the similarities between objects that are functionally equivalent, or even identical.

If similarities between objects are to be reflected in their descriptions, the representation must be capable of expressing shape in a way that is

independent of surrounding objects, and independent of the viewing position. This can be achieved by describing objects in terms local to the object itself. The head of a bolt, for instance, might appear completely different when seen on the side of a car under assembly, than when seen on a bench with a collection of ball bearings. The functional shape of the bolt head is identical in different situations, however, and it should therefore be described in local terms, relative to the rest of the bolt, rather than relative to the scene in which it appears. If the heads of bolts are always described identically, the invariance of the head's functional shape is easily recognised.

In reasoning about interaction between objects (for instance, analysing fasteners), it is necessary not only for the *objects* to be expressed in local terms, but also the *relationships* between the objects. For example, a split pin in a vehicle assembly always works the same way, whether it passes through an axle, or through a steering link. In this case, the overall shape of the axle or link is irrelevant, and it is only a few features near the split pin that are functionally important. It should be possible to represent a fastening situation such as this in terms of the local features, the orientation of those features relative to each other, and the motions required to act on them.

Three aspects of object representation which can be formulated in terms of a local context are size, location, and orientation. Size can be described in local terms by comparing feature sizes either to the overall dimensions of a complete object, or to distinctive features in the vicinity (using a local quantity space in qualitative systems). Location and orientation can be described relative to distinguished points and directions in the remainder of the object shape, rather than with respect to global axes. Using the object itself or an interacting object to provide a reference frame means that descriptions of similar objects or interactions will always be made in consistent terms.

Local orientation must be described by reference to distinctive shape features. Candidates for orientation reference features include object axes, or dominant directions. Dominant directions can be calculated from boundary representations, where dominance is established by the total length of edges and/or surface normals with a particular orientation. Object axes are usually specified in constructive solid geometry representations, where each primitive has intrinsic axes. Both of these directions are functionally significant in many mechanical parts (in fact, the directions often coincide in mechanical parts), and the representations discussed later in this chapter make use of both.

Local context descriptions can be made relative to a reference frame derived from the main object, or derived from neighbours of the object, where interaction between objects is particularly significant. Another possible reference frame for local context is provided in a representation with multiple levels of detail, where a coarsely described shape can be used to provide the local reference frame with respect to which the details that compose it will be defined. One of the representations described later in the chapter does precisely this – in that representation I refer to the coarse feature reference frame as an “imple context”, because complex concave or convex features of an object’s shape can be described at a coarse level as a dimple or pimple on the object. Size, location, and orientation of detailed shape features are then described relative to the imple.

In addition to static descriptions of object shape, size, and orientation, motion of objects can be expressed in terms of local contexts. Where two objects are joined together by small fasteners, the overall motions of the objects for unfastening are best described by reference to the shape of the fasteners. If the fasteners are described using imple contexts, then the motion of the complete object should be described in terms of those contexts. The extensive use of local contexts therefore requires that transformations be available between local and global coordinates, as described by Popplestone, Ambler and Bellos in [PAB80], and by Ballard [Bal84].

This section has described the benefits arising from shape description in terms of local shape context, and has discussed some mechanisms which can provide these abilities. The qualitative representations presented later in the chapter make use of qualitative adaptations of these mechanisms.

4.1.3 Representation of Multiple Features Using Functional Groupings

Robots usually perform repetitive tasks, and an intelligent robot should be able to recognise and take advantage of situations where the same objects or object elements are encountered more than once. Such capabilities are an important element of most CSG solid modellers, where they are the equivalent of macros or subroutines in programming languages. If, for example, a table is being defined using a CSG scheme, a single leg might be described as the union of a truncated cone and a disk (at the foot). This description, instead of being repeated three more times at other corners of the table, could be defined as a functional grouping with the name “leg”, allowing a more straightforward description of the table by referring simply to a leg at

each corner.

The ability to define a complex shape as an abstract functional grouping enables a robot to associate specific properties with the whole group of features comprising that shape. There are a number of properties which could be usefully linked to abstract shape groupings, as follows:

- The essential elements that define the functional grouping must be recorded. Any given grouping may have more complexity than necessary to belong to the group, but this list of necessary properties ensures that it will behave as expected.
- Deductions that have been made in the past concerning properties of equivalent groups can be recorded, so that information can be re-used.
- Manipulation strategies employed to perform operations on the group can be stored, together with their relative success.
- Associated shapes or groups can be recorded, so that the functional description of a `nut`, for instance, can be linked to that of a `bolt`.
- The inclusion of variations between successfully matched group members provides a historical basis for generalisation from examples. Some negative information regarding unsuccessful partial matches would allow even more power in generalisation.

Similar advantages are listed by Noah and Sacerdoti as arising from the addition of “macro operators” to a robot planning system. MACROPS enabled the planner to collect effective sub-plans so that they could be re-used in appropriate situations, and functional groupings provide an analogous mechanism which records physical descriptions for re-use.

The two representations discussed later in this chapter have not been extended to allow all of the above capabilities in matching and collecting information about functional groupings of shape elements. The suitability of each of the two dimensional representations for functional grouping description is discussed, however.

4.1.4 Qualitative Representation of Size

Reasoning about operations on physical objects requires some way of expressing the sizes of objects, the sizes of object features, and distances between objects; this is the aspect of scene representation in robot reasoning

systems which normally involves a large amount of quantitative information. The quantitative information available is essentially a set of measurements of the scene. For a qualitative representation to be useful, some size information must be retained, while reducing the reliance of the representation on numeric information.

What kind of size information should be retained in a qualitative representation for use by a robot? The alternatives include a quantity space with distinguished points, such as that used by Forbus [For84a], a set of fuzzy values, as defined by Zadeh [Zad79], a combination of the two techniques as proposed by D'Ambrosio [D'A87], or a set of relative sizes.

Typical tasks involving size judgement which a robot might encounter are:

- There are two parts A and B. Part A has a hole in it. Will part B fit into the hole?
- Past operations have involved screwing a screw A into a hole in workpiece B. Screw C is slightly different to A. Will C fit into the hole in B?

In both of these robot tasks, it is relative sizes of the workpieces that have functional significance in the context of the task rather than their absolute or measured size.¹

The size of object components can be described relative to characteristic dimensions of the overall object. In two dimensions, for example, there are two characteristic size references which are easy to determine for any object. These are the widest extent of the object (along which I define the “major axis” in the first two-dimensional representation described below), and the narrowest extent (the “minor axis”).

If some characteristic size is chosen as a local size reference value, it could be used as a distinguished point in the size quantity space, so that qualitative size distinctions are made by comparison to this value. There are however two disadvantages to such an approach. One of these is that there are qualitative size distinctions which cannot be made in terms of the reference. If a bolt were described by reference to its length and the width of its head, there would be no way of discriminating between the thread

¹Absolute size information may become important when the robot is physically operating on workpieces, if positioning and motion of robot arms are specified as absolute coordinates within the workspace, but it is not always necessary during task planning stages.

diameter and thread pitch. The second disadvantage is that the reference sizes may have no relevance at all to the operation being planned. If a collection of bolts are being matched to appropriate nuts, the length of each bolt is quite incidental.

The first of these disadvantages can be alleviated by further dividing the quantity space, in order to discriminate between sizes that are smaller than any of the reference values. This can be done by defining fractions of the reference values as new “distinguished” points. One of the representations described below uses this approach, and simply defines binary orders of magnitude as appropriate fractions, so that something more than half the size of the reference value can be contrasted with something less than quarter the size.

The second disadvantage is more difficult to relieve, and is actually exaggerated by the binary orders of magnitude approach, because of the effect of surplus distinguished points in the quantity space. These extra points can arbitrarily produce a perceived distinction between close values that are actually qualitatively equal, but happen to fall on either side of a binary order of magnitude point. The object-relative size representation described below does not attempt to solve this problem, but other researchers have attempted to do so – notably Connell and Brady, who used a Gray coded representation for the ordering of feature size ranges when measured as a proportion of object axis size [CB87]. The use of Gray coding allowed values to fall within overlap regions, where they share the properties of both neighbouring size ranges. Alternatively, magnitude comparison can be performed on a “fuzzy” basis, as described by D’Ambrosio [D’A87].

There are numerous ways of extending the local reference value approach so that the size criteria used are less arbitrary than simple comparison to overall object size. One of these is to allocate special values according to a statistical analysis of the distribution of feature sizes in the scene. It would then be possible to describe the sizes of individual shape features by reference to these statistically compiled values in the same way as for comparison to axis sizes.

Another approach to size comparison is to use some simple technique to broadly distinguish size across the whole scene, then provide a size-on-enquiry ability to determine relative sizes of objects only when that information is needed. This allows an initial simple distinction to be refined if necessary. Such an approach avoids the accumulation of superfluous and even misleading size classifications, but it does incur computational overhead, since a program acting on a scene description must decide when it is

necessary to find further information.

Relative size representation is the first facility to be considered in designing a qualitative spatial reasoning system. It provides a natural way to describe size dependent operations without reference to any specific measurement system, and it provides facilities that can operate with inexact data. There are a number of options to follow in achieving relative size description. This type of capability is an immediately apparent requirement of a “qualitative” spatial reasoning system, since linear measurement information is the obvious candidate for application of the quantity space. The first of the two representations presented later in this chapter uses a system of distinguished points in the size quantity space that are derived from local shape axes, while the second uses a partial ordering of size on a global basis.

4.1.5 Summary of Qualitative Representation Issues

In this discussion of four spatial representation issues I have identified a number of capabilities that are desirable for a qualitative spatial reasoning system if it is to be used by a robot, and I have also suggested ways of providing those capabilities.

Multiple levels of detail in shape description allow sensory data to be organised at a high level of abstraction, while retaining the information from which the abstract description was constructed. This gives a reasoning system access to all relevant information when the derived high level description is insufficient.

The way in which people make use of the ability to consider complex shape at multiple levels of details can be emulated in two ways: a complete description of each scene can be stored, with index links to appropriate parts of the high-level description. Alternatively, an initial coarse description can be constructed, with the reasoning system directing focus of attention to obtain more detail where necessary. The representation techniques described in the rest of the chapter can support both techniques, by allowing indexing either from coarse to fine levels of detail, or vice versa.

Many human spatial reasoning tasks depend on the ability to consider features or operations in a specific local context. This ability can be used either in concentrating on one portion of a very complex overall shape, or to apply similar operations in similar local contexts, regardless of variations outside the local context.

I describe a specific method for local context shape representation, where an “imple” is a portion of overall shape that can be treated as an object on

its own. The imple defines local context for size, location, and orientation. Imples also provide a mechanism for description at different levels of detail – the imple context may contain detail that was not appropriate to include at the level of detail description where the imple shape was defined.

The human ability to assign properties to an abstract functional grouping of shape elements provides a number of capabilities that would be useful to a robot spatial reasoning system. These include a record of successful deductions and manipulation strategies, together with a basis for generalisation and identification of previously encountered situation types.

The following representations do not include this facility, but both were designed with consideration to how functional groupings of shape elements might be achieved.

Relative size description is the most basic facility for a qualitative spatial representation. It is the normal mode of operation for people, who seldom need to measure objects before operations can be performed on them. A variety of techniques for relative size description have been proposed, and the two representations discussed below make use of two different techniques – one an axis-relative quantity space, and the other a scene-global size ordering.

4.2 2D Qualitative Geometry from Solid Modelling

My initial approach to qualitatively representing objects and their interactions and relationships was to adapt and extend commonly used solid modelling techniques, so that they could be implemented using qualitative methods, while meeting some of the requirements discussed in the previous section. Using solid modelling techniques resulted in two different approaches to the problems involved: one developed from boundary representation methods, and the other from constructive solid geometry methods. Although my initial investigations recognised that any development aimed at robotics should take a three dimensional approach, both of the modelling methods involved sufficiently complex problems in two dimensions that my implementation was restricted to that case, and I made no attempt at implementing a three dimensional representation.

Following relatively independent paths in developing these two representations has resulted in interesting differences between the two. Both two dimensional methods will therefore be described in detail below, although more space is devoted to the one that I conclude is superior. As an intro-

duction to the two dimensional shape representations, this section describes the way that they were derived from three dimensional representations.

4.2.1 A Two Dimensional Derivative of Constructive Solid Geometry

The basis of constructive solid geometry is the description of complex shapes as a combination of simpler ones. Therefore the fundamental requirements of a CSG scheme are: a set of primitive shapes, and a method of describing the relationships between primitives. Applications of CSG in the robotics domain (e.g. [WLPL⁺80]) had previously found that it was also useful to be able to define new primitives, such as solids of revolution, or laminar shapes.

A large set of simple primitives for CSG, and also some complex shapes, can be elegantly defined using the generalised cones method. Geometric primitives such as cones, pyramids, or blocks can all be described as a two dimensional shape swept along an axis, as can mechanical “primitives” such as washers, brackets, channels and wires. This appeared to provide a consistent way of both describing standard primitives, and defining new ones when necessary. The sweeping axis can also be used as a reference direction when combining primitives.

There is no two dimensional equivalent of the generalised cones method that provides the same kind of power in description of physical objects. It is certainly possible to construct two dimensional shape in terms of swept line segments, but there is little advantage to be gained in doing so, especially since it is difficult to gain qualitative knowledge about the shape from this type of construction. The alternative that I chose was to define shape primitives in terms of their qualitative features – features such as curves, angles, and lines.

The shape elements that are constructed from these features are still defined in terms of axes, and are combined in the same way as subobjects are combined in CSG schemes – by defining the relative positions of their axes, together with a logical operator. A single general method was used to define objects, subparts and features; all shape features are described in terms of their axes, so that the feature can be described relative to the overall axes of the subpart, which are then described relative to the axes of other subparts.

A conscious concern in proposing this representation was that it should be feasible to construct a scene description from visual data. This is one of the disadvantages of constructive solid geometry – although it is easy for

people to define a complex shape using CSG, it is not so easy to take a picture of a complex shape, and construct a CSG representation from it. In fact, CSG cannot be used as the basis for a canonical shape description, because there is no unique set of primitives which combine to form a given shape. The feature-based qualitative representation could however make use of a number of techniques for feature detection in scenes (such as [KJ86] [RKH85]).

Connell and Brady describe a method for decomposing two dimensional shape into CSG-like primitives. The smoothed local symmetry representation makes use of cues from the object boundary to find “joins” that break an object into subparts [CB85a]. I have made use of a far less sophisticated technique for identifying joins: subparts are defined as being convex, and the junction between two subparts is therefore delimited by a “waist” – the narrowest extent between concavities. A canonical description of two dimensional shape is provided by the smallest possible number of convex subparts.

The relationships between these subparts can be described with respect to axes. I define the “major axis” of a shape as the line along which it extends furthest, and the “minor axis” as that along which it is narrowest. These axes are used to describe relative position of subparts, and are also used as size references. When an overall shape is divided into convex subparts, these are also defined and oriented by their axes. The convex subparts are described as collections of shape features, all of which are related in terms of their own axes’ orientation with respect to the major and minor axes of the subpart.

This description of subparts and shape features in terms of the relative positions of their axes I describe as “Axially Specified Subparts and Features”. For convenience, I will refer to the general method by the initials ASSF.

4.2.2 A Two Dimensional Derivative of Boundary Representation

The second approach which I took to two dimensional qualitative geometry was developed from boundary representation methods rather than from constructive solid geometry methods. The reason for this was that the most important aspects of the ASSF representation in reasoning about motion seemed to be boundary related rather than axis related, as I explain later.

Solid modelling using boundary representation requires a larger amount

of information to describe basic three dimensional shape than the combination of constructive solid geometry and generalised cones does, but it can also describe a wider range of objects. This is because all surfaces of the object are described explicitly, whereas in a CSG description they are implicit in the combination of primitives, and in the shape sweeping methods used for describing primitives.

Explicit boundary description provides important advantages where the representation is used for reasoning about interaction between objects, rather than just properties of a single object. This is because objects only contact other objects on their boundaries, so a description of the boundary must be available to the reasoning system, whether it is given explicitly, or determined by computation from a constructed solid description.

The generalised cones method requires that a method of representing two dimensional shape be used to describe the cross section for “sweeping” operations, but a boundary representation must describe the two dimensional shape of every face in the three dimensional object. This has the advantage that individual features are separately described in local two dimensional contexts for the whole object, whereas for CSG, features in planes other than the cross section must be inferred from the sweeping function.

An object boundary can be qualitatively described in two dimensions by identifying sections that are qualitatively homogeneous, then describing the relationships between those sections. If the homogeneous sections were all straight lines, the resulting description would be a polygon. For more generalised shape description, the sections can also be curves or wiggles, and the description becomes an extended polygon.

The qualitative representation which I derived from three dimensional boundary representation describes shape in exactly this way – as a collection of qualitatively different segments arranged to form an extended polygon. For brevity, I will refer to this extended polygon boundary method as EPB.

4.3 Qualitative Two Dimensional Shape Description

This section describes in detail the two systems for two dimensional shape representation which developed from the approaches to solid modelling discussed above. I have used both of these systems to construct descriptions of example shapes and scenes. Chapter 5 describes programs which operate on both representations, and discusses the relative advantages of each one.

For each of the two approaches, I will discuss the methods used to represent four aspects of qualitative shape: primitive shape, relative size, object-interior position (the relative position of subshapes within an object), and object-exterior position (The relative positions of complete objects, or of parts on different objects). These four are sufficient to describe any combination of objects in a scene, and any combination of subshapes within an object.

4.3.1 Describing 2D Shape with Axially Specified Subparts and Features

The axial specification method provides the following basic shape description mechanisms:

- Shape is described in terms of a combination of distinct convex subparts, and two dimensional shape features.
- Relative size is described in terms of the lengths of object axes, subpart axes and feature axes.
- The position of subparts and features within an object is described with respect to the object axes, subpart axes and feature axes.
- Relative position of separate objects is described in terms of contact between features or subparts, and relative orientation of object axes.

The most prominent aspect of this two dimensional shape representation method is the use of axes to describe both size and orientation, and that every element of the shape description has axes associated with it. The axes associated with important shape description levels are as follows:

Overall Shape: The **major** and **minor** axes. These were described above as the widest and narrowest extents respectively of the shape.

More precisely, the **major** axis is the longest perpendicular line that can be drawn between two parallel tangents to the shape boundary, and the **minor** axis is the shortest such line. “Tangents” may span vertices, so that the height of a triangle, for instance, could become an axis by this definition.

Convex Subparts or “Imples”: The major and minor axes, and the waist. Major and minor axes of imples are defined as for overall shape, with

the waist being considered part of the **imple** boundary during axis construction.

Vertices: The angle bisector.

Edges: The chord between edge extremities, and the normal to this chord.

Primitive Shape

The qualitatively different types of shape feature which can be combined into an overall shape description fall into five major categories:

- Straight edges (**edge**).
- Vertices (**vertex**).
- Simple curves with no points of inflection (**curve**).
- Complex curves with multiple points of inflection (**wiggle**).
- “Dimples” and “pimples”, which are represented in the same way as a convex subpart is. When viewed at a higher resolution, they are composed of smaller features. The axes of an **imple**, and its relationship to the object, are defined as for a convex subpart.

Each of these categories is then further divided in ways that are either object independent, or relative to axes of the object or of other features. Straight edges are described only according to relative size on an object scale, (this will be discussed in the size representation section), but there are shape subcategories for all the other primitive shape feature types, as follows:

Vertices are described by the **angle** of the vertex, which can have one of seven qualitative values. These seven include four value ranges, separated by three distinguished points. The distinguished points are based on the right angle, two being the **concave-right** and **convex-right** angles, and the other the **straight** angled vertex, which can be used to describe points where two features meet at an angle of 180° (e.g. a curve and a straight edge). The four ranges of values between these points are described as **concave-acute**, **convex-acute**, **concave-obtuse** and **convex-obtuse** angles.

Simple curves are described by size in the same way as straight edges, but they are also described by the angle through which the curve turns,

and by the spread of the curve. The same qualitative angle values are used for curves as for vertices, with the exception that a straight angled simple curve is not described as a curve, but as a straight edge. This allows the description of concave or convex curves, as well as of varying amounts of curvature. Spread is expressed in terms of the ratio between the length of the chord axis and the maximum deviation of the arc from that axis. This deviation is the **bulge** of the curve. A simple set of qualitative values uses circle shapes as distinguished points in the quantity space, because the circle is a curve with many useful geometric properties. A **semicircle-like** curve has a **bulge** that is half its chord axis, and a **circle-like** curve has a **bulge** that is equal to its chord axis. Three qualitative ranges are separated by these two distinguished points, so that there are five possible qualitative values for curve spread.

Complex curves (**wiggles**) are described according to the number of points of inflection, and the amplitude of the **wiggle**. The number of points of inflection need not be exact, with a distinction between **few** or **many** wiggles being sufficient to indicate the qualitative nature of the shape. Amplitude (or **bulge**) can be defined in a manner that is similar to that used for simple curve definition. My formalism divides **wiggle** amplitude into qualitative ranges between a quarter and half the length of the chord axis. This is more arbitrary than the classification of simple curves and vertices, however, since it is not based on geometric properties (such as right angles or circles). A less arbitrary basis for classifying amplitude might be desirable, but a complex technique is unjustified because wiggles are not common in the mechanical domain, and I avoid them in my examples.

The major categories of **imple** are convex (**pimple**) and concave (**dimple**) shapes. There is a lot of variation in shape between **imples**, but this is reflected at a deeper level of detail, where the **imple** is described as if it were a separate subpart. An **imple** has two pairs of axes: The first pair are the **waist** line separating the imple from other convex subparts, and the normal to that line. The second pair are the **major** and **minor** axes with respect to which the imple shape is defined. The use of two pairs of axes means that the imple can be regarded as a curve when its exact shape is not important (the waist axes can be regarded as curve axes), while the complete shape description can be related to that curve model for operations on the imple itself.

Relative Size

Relative size can be represented qualitatively at the simplest level by the qualitative relations: bigger, smaller, and equal to. In the ASSF system, relative size of objects within a scene, features within subparts, and subparts within objects, are all expressed in terms of the lengths of different types of axes.

The two axes which are most often used as a reference for relative size are the **major** and **minor** axes of each distinct shape; such shapes can be either a complete object, a subpart, or an imple. Local relative size is expressed by comparing a given dimension to these axes. Five qualitative values are immediately available: there are three qualitative ranges (less than the minor axis, between the minor and major axes, and greater than the major axis) and two distinguished points (**major** and **minor**).

Section 4.1.4 above points out the ways in which this approach is unsatisfactory, and also some partial solutions. One partial solution is to further subdivide the quantity space, so that it is possible to make qualitative distinctions between sizes that are closer in magnitude. I have done this by introducing further “distinguished” points at one half and one quarter of the major and minor axis lengths. This results in a quantity space with six distinguished points and seven qualitative ranges, which can be used to construct a satisfactory representation of quite complex objects (especially where local complexity is treated using a local set of imple axes), but which still suffers from the limitations described in section 4.1.4 if there are essentially equal sizes that are accidentally separated into different quantity space ranges.

Individual features are measured along their own axis, which is usually a straight line between the points at either end of the feature. This is true of straight and curved edges, and also of imples, which are measured along the waist where the imple meets another convex part. Secondary axis measurements are used to define the shape of an imple, by specifying the size of imple major and minor axes relative to the axes of the main shape. All of these feature sizes are described by comparing them to the local quantity space defined from the major and minor axes of the main shape.

Object Relative Position

The object relative position of a feature describes where it is with respect to the complete object. Position is defined in terms of the location of the

feature and its orientation. Each of these is described in two ways: relative to the major and minor axes of the whole object, and relative to other features.

The location of a feature with respect to the whole object is described in terms of a reference point on the feature. The reference point for an **edge**, **curve**, or **imple** is the centre of the chord or waist axis, and the reference point for a **vertex** is the vertex point. The position of this point in a two dimensional space defined by the major and minor axes of the object is described by stating which end of each axis the point is nearest to, and how near it is.

The location of a feature with respect to other features is described by the order in which the features appear when the boundary of the object is being traversed in a clockwise direction. This description is independent of the overall shape of the object, but features which cannot be directly related to the boundary do not have any description of this type.

Orientation relative to the overall shape is specified in terms of a reference direction derived from the axes of individual features. The reference direction for curves and straight edges is the normal to the chord axis, the reference direction for vertices is the vertex axis, and imples are multiply defined in terms of several reference directions: the normal to the imple waist, the major axis of the imple shape itself, and the minor axis of that shape. The orientation of these feature reference directions with respect to the rest of the object can be described by a qualitative direction space oriented to the intersection of the major and minor axes. This qualitative centroid provides an origin by which four quadrants around the object centre can be defined. These four quadrants then produce a range of qualitative direction values with four distinguished directions, and four value ranges.

Orientation with respect to other features is only specified where there are special relationships between two features, since relating each feature to every other feature would become very complex for more than a few objects. The particular relationships defined are those between **parallel**, **aligned**, or **perpendicular** axes. Lists are maintained of every set of features in a scene which are parallel or aligned; where two of these sets are perpendicular to each other, that is also noted.

Object Position

Object position is a description of the orientation and location of complete objects with respect to other objects in the scene. This description is

achieved in two ways: firstly in terms of the object axes, and secondly in terms of features that are near each other.

Relative location of two objects is expressed by reference to the intersections of major and minor axes on each object. The distance between these two points is measured in terms of axis lengths, choosing whichever of the main axes of each object is an appropriate measure. The second component of location is the angle relative to each object at which the other intersection is found. This is expressed by dividing the space around the object into four “quadrants” (which may or may not meet at right angles) divided by the major and minor axes. The angular location of the other object is then described by the quadrant in which its axis intersection falls.

Relative orientation of objects is described in the same way as relative orientation of features on a single object; where axes of the two objects are **parallel**, **aligned**, or **perpendicular**, this is noted. No effort is made to express exact angular orientation at any other angle, but qualitative orientation can be ascertained by comparing the relative angular location from the point of view of each object.

The relative orientation of features on different objects is described in two different ways. The most functionally important one specifies where objects are in contact: the names of the two features that are in contact are recorded on a global list of all contacts in the scene. The other records sets of parallel, aligned, or perpendicular straight edges. These sets are maintained in the same way as for features on the same object, and for aligned object axes.

Axially Specified Subparts and Features Summary

An implementation of the ASSF representation for two dimensional scenes is presented in the next chapter, which describes a reasoning system that uses this representation. During the construction of that system, the following advantages and disadvantages of the representation became apparent:

Advantages:

- Predicting interaction between objects requires that the arrangement of features around the object boundary be explicit. Lists of features in order around the boundary were extensively used for this purpose. An explicit neighbourhood pointer for each feature would have been even more useful.

- Contact between objects can be easily represented by simply listing all features on the objects which touch each other.
- Hiding detail by the use of imples allows gross motions of objects to be planned without considering small details of the object shape.
- Convex subparts provide a simple and consistent method for dividing two dimensional shape into simpler shapes. In addition to providing a computationally feasible technique for acquisition of shape description from visual data, it is close enough to design techniques such as constructive solid geometry that the representation is clear to humans.
- The use of object axes as the basis for a representation can provide some degree of functional information about the object. This is evident in the “Mechanic’s Mate” project [BA84a], where shape axes are used as cues to mechanical function; they can indicate possible rotational motion, linear motion, or direction of force application.
- Overall, the ASSF shape description method seems reasonably close to human techniques for describing shape; it is possible for a person to reconstruct the shape of an object given a description in this form.

Disadvantages:

- Defining size with object axis length as a size reference does not assist reasoning about object interaction, because it obscures the relative sizes of shape features on different objects (since shape features on each object are related to the axis on that object only). Reference to axis length would be more useful in solving problems in CAD, object classification, or other areas where objects are considered one at a time.
- Description of the size of shape features by comparison to the major axis fails to recognise the common case where two shape features fall within the same subdivision of the axis size, but have a functionally important difference between their own sizes.
- Although object axes can provide functional information, local shape features on the object boundary are far more significant where close interaction between objects is occurring, and the relative orientation of features is therefore more important in this situation than axis orientation is.

- The methods for describing location and orientation of shape features are rather clumsy. They describe location and orientation sufficient that object shape can be largely reconstructed (by a human) from the description, but they are difficult to use when reasoning about motion is being carried out.

4.3.2 Describing 2D Shape With Extended Polygon Boundaries

The second of the two representations I have developed describes object shape in terms of the boundary of the object. This representation provides the four basic shape description mechanisms introduced at the start of the previous section in the following ways:

- Primitive shape is expressed at the level of boundary elements, which are either qualitatively homogeneous segments of the boundary, or the junctions between those segments.
- Size is described using a partial ordering of the distances between objects and the lengths of boundary elements.
- The positions of boundary elements local to each object are described in terms of the order in which boundary elements appear on the boundary.
- Relative positions of separate objects are described in terms of proximity between boundary elements on each object.

This shape representation does not make any use of axes, and it provides no space filling information of the type that is provided by the subparts or imples of the ASSF representation. The information which it does contain is in fact mostly a subset of the ASSF representation – a description of the shape boundary can be constructed using the types of qualitative shape feature described as part of the ASSF representation. As a subset, the basic shape representation of the EPB technique is therefore simpler, but less intuitively obvious to humans than the hierarchy of subparts and features.

Primitive Shape

The basic components of the extended polygon boundary are “segments”, and “junctions” between segments. A **segment** is a qualitatively homogeneous length of boundary corresponding to an edge of the extended polygon.

Element Type	Shape	Attributes
segment	line	length
	curve	length angle concave/convex bulge
	wiggle	length period bulge
junction		angle concave/convex

Table 4.1: Boundary Element Types

A **junction** is the place where two homogeneous lengths of boundary meet, corresponding to a vertex in the extended polygon. Segments and junctions are both types of boundary “element” (in fact, all boundary elements are either segments, or junctions).

The term “qualitatively homogeneous” is used to describe either a straight line segment, a curve without discontinuities, or a **wiggle**, which is an untidy piece of boundary that is differentiated from other tidier segments by transitions at each end.²

These three types of boundary elements were derived from the shape features described above as part of the ASSF representation. Table 4.1 summarises the types of boundary element that are required to describe a wide range of 2D shape in a mechanical domain.

The range of values which can be assumed by the **angle**, **bulge**, and **period** attributes are the same as those used in the ASSF representation, and were chosen for the same reasons given in the previous section.

The EPB representation supports multiple levels of shape detail with a mechanism for defining one or more simple elements as an alternative to

²These three types of boundary segment were selected primarily because they were convenient in motion planning problems, but an advantage in this selection is that it is possible to extract this information directly from a visual image, as shown by Mackerras [Mac87b]. His system produces an “RSK” segmentation graph, containing “regions”, “segments” and “knots”, which is similar to my extended polygon representation. Brady’s Smoothed Local Symmetries vision system [BA84b] also produces shape description output in this form.

complex portions of the boundary.³ Operations which do not need exact data can then deal solely with the simplified boundary elements where it is appropriate. It is possible to nest these parallel representations, so that an overall shape can be progressively simplified.

Parallel representation can also be used to provide alternative descriptions of single shape features. For example, a knife with a serrated edge can be treated the same as a knife with a straight edge for most operations – the serrations are only a significant detail when the knife is sliding along that edge. The parallel representation can be used in this case to describe the knife-edge segment of the knife boundary as a **line** on the coarse branch of the parallel description, and as a **wiggle** on the fine branch.

The same approach can be used to describe manufacturing features such as fillets or chamfers. A curved boundary segment with a small radius can be described at a coarse level as a simple **junction** between the segments on either side of it. The transformation between **curve** and **junction** retains the **angle** of the curve in the junction description, and discards the curve size.

Relative Size

The Extended Polygon Boundary representation expresses size in relative, rather than in absolute terms, the same as the ASSF representation. The ASSF system described the size of all features in a single shape relative to the length of that shape's axes. The disadvantage of this system was that it was not possible to directly compare the sizes of two features on different shapes. In fact, it was not always possible to compare the sizes of two features on the same shape, if they fell into the same range between distinguished points in the length quantity space.

This problem can be avoided by describing boundary element sizes relative to each other, rather than relative to a few reference values (whether axis lengths, or other values). If relative size is recorded on a global basis, rather than local to one shape, it is also possible to compare the size of elements on different objects. This ability is essential for reasoning about most types of object interaction.

Recording the relative sizes of all boundary elements in the scene results in a global size ordering. The use of a complete global ordering would provide a large amount of information, but removes some of the advantages of a

³See Appendix A for a LISP code example of the way that multiple levels of detail are represented on an object boundary.

qualitative representation, in that exact measurement information is needed to construct the ordering. In addition to this construction requirement, the ordering must be partially re-constructed when any movement occurs in the scene. The maintenance of a complete ordering would require that some numeric information be included for re-calculation of the ordering.

This problem can be overcome in part by using a partially ordered space instead of a complete ordering. A size value is given a context in the partially ordered space by three links: one to any other size that is known to be equal in magnitude, one to the smallest size in the space that is known to be larger than it, and one to the largest that is known to be smaller. The resulting partial distance ordering is a network of size relationships similar to the “quantity lattice” used by Simmons [Sim86] in investigating qualitative arithmetic.

The greatest advantage of the partial distance ordering is that exact information can be represented together with uncertain information. If all sizes in a scene are known precisely, a complete ordering can be created, while if no information at all is available, all sizes in the scene can be represented in parallel as equally uncertain. A normal application, of course, will involve a mixture of certain and uncertain information, that is represented as a network.

A typical situation where this is necessary is motion planning, where static measurements of the scene before any motion occurs are exactly known, but intermediate positions of the moving object during the proposed motion are only estimated. These estimated positions can be described as a limited range of possible values, while the exact information retains a strict ordering.

Techniques for reasoning with a partial ordering can make use of as much information as is available about any given measurement, thereby providing a natural graceful degradation of system performance when less precise data is available. The implementation described in the following chapter behaves in exactly this way, and it is able to continue reasoning either with imprecise input data, or after modifying its internal representation with inexact qualitative operations.

There are two types of distance information which must be available to a spatial reasoning system: sizes of things, and distances between things. In the EPB system, both types of information are recorded in the global partial distance ordering. The size of boundary segments is recorded explicitly as a length attribute of the segment, which points into the ordering. Distance between boundary elements, which is called “proximity”, is recorded as part

of a link between the elements. Proximity can be used to describe the distance between two elements on the same object, in which case it might also convey implicit information about the overall size of the object, or it can be used to describe the spatial relationships between two different objects.

The implementation of the partial distance ordering uses “equidistance lists” – sets of distances (which might be either segment lengths or proximities) of equal magnitude. The zero point in the ordering is the **contact** list which is the set of boundary elements that are in contact with another element. Contact is therefore treated the same as any other proximity – the distance simply has a magnitude of zero.

Object Position

The EPB representation contains no explicit information about overall object shape; it only has information about elements of the object boundary. This means that any description of the relative positions of objects must be in terms of their boundary elements rather than their axes, for example. This is not necessarily a disadvantage for purposes of analysing object interactions, since any interaction between the objects will take place on their boundaries. The position of an object in terms of its boundary elements can be defined relative to all other objects in the scene simply by stating what other objects have proximity relationships to each boundary element. This proximity information in the partial distance ordering constitutes an effective qualitative representation of two dimensional position.

Since boundary elements are used as the basic unit for describing the overall scene, rather than object axes, as in the ASSF representation, some technique is necessary for reducing the overall number of relationships described. This is because the total number of possible relationships between all boundary elements in the scene is the square of the number of elements in the scene, and becomes very large for even a moderately complex scene.

The method used to reduce the number of relationships is as follows: proximity is only measured between those boundary elements which would come into contact if the objects were to move toward each other. A simple transform can be carried out on visual scene data that filters out unlikely contacts, and also orders the remaining element proximities appropriately for inclusion in the partial distance ordering.

The proximity transform involves simultaneously “expanding” the boundaries of every object in the scene by constructing an enclosing outline around it. The boundaries are expanded in steps until elements on the expanded

boundaries come into contact (this process is shown in Figure 4.1). The set of contacts that occur at a given expansion step are formed into an equidistance list, and these lists are ordered by the stages in the expansion at which they are formed. This information is integrated in the partial distance ordering with object internal proximities and segment lengths by simultaneously “expanding” along segments and inside objects, so that these distances can be formed into ordered equidistance lists together with proximities between boundary elements on different objects. This allows direct comparison of segment length with distances between objects.

The combination of boundary specified object relationships and parallel levels of detail on the object boundary means that at coarse levels of detail, solid objects can appear to interpenetrate. Consider two combs which are interlocked, as shown in figure 4.2. When viewed at a fine level of detail, the teeth of the combs are seen to be laced together. When viewed at a coarse level of detail, however, the combs have no teeth. The toothed edge of the comb may appear to be a **wiggle**, or even a straight boundary **segment**. This is a reasonable representation, since many operations can be performed on the comb as if the teeth were a straight edge (such as picking it up by the edges), however the apparent overlap of the combs now means that the proximity between the two coarsely represented edges should be negative, and this is inconsistent with the “size” ordering, which is a representation only of magnitude.

This apparent boundary overlap is a common situation when fastening devices are being considered. Fastened objects are usually interlocked in some way, and the interlocking mechanism is often small by comparison to the whole object, so that at some levels of detail the object boundary will simply be extended to include it. The use of a “negative” proximity here, although not strictly meaningful in a magnitude ordering (magnitudes are always positive), allows the amount of overlap of the coarse boundaries to be explicitly represented. To allow this, overlaps are included in the partial distance ordering like any other proximity information, but they are tagged as a special type. This means that the complete partial distance ordering may include four different types of distance: length, **internal** proximity, **external** proximity, and now **overlap**.

Object Relative Position

The position of individual boundary elements relative to the whole object is represented using two types of information. One is the order of elements

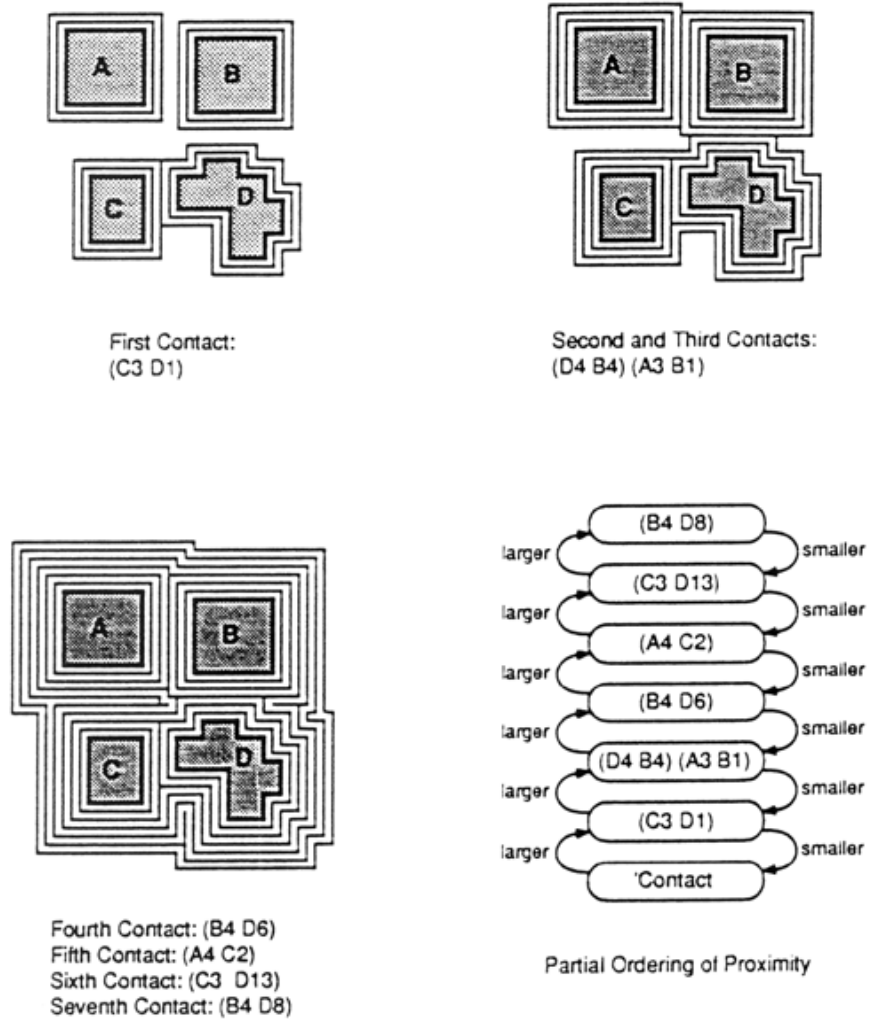
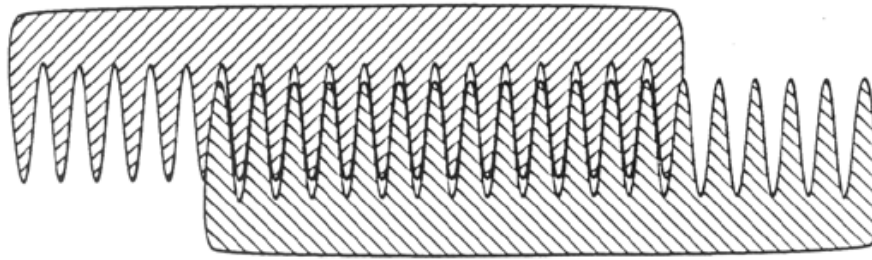
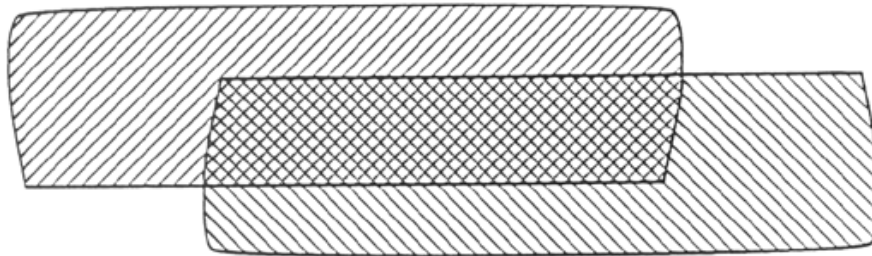


Figure 4.1: Proximity Measurement in Terms of Boundary Expansion



Interpenetrating Objects at Fine Level of Detail



Apparent "Negative Proximity" at Coarse Level of Boundary Detail

Figure 4.2: Apparent Object Interpenetration at Coarse Levels of Detail

around the boundary, and the other is the record of any places where an element is in proximity to another element on the same object, as determined by the proximity transform described in the previous section. The proximity transform, in which the object is also “expanded” with outlines inside the shape boundary, produces a canonical distance ordering for object **interior** distance that is consistent with the description of **exterior** distance. Note that the transform does not have the same interpretation for motion planning in this case, since there is no possibility that the boundaries of a rigid object will come into contact.

Internal proximity such as this is recorded in the partial distance ordering, with a tag to indicate that the elements are part of the same object. Boundary order is represented with a pointer from each boundary element to its **left** and **right** neighbours. The resulting representation of the boundary is a doubly-linked circular list, containing alternate **segments** and **junctions**.

This description of the boundary representation is slightly simplified, in that it omits the mechanism for multiple levels of detail on the object boundary. It is possible to represent sections of the boundary in multiple ways by allowing a number of parallel alternative sections of boundary. To achieve this, each boundary element does not have just a single pointer to its neighbours on the **left** and **right**, but a list of pointers, ranked in order from elements which are part of the coarse detail to those which are fine detail. The circular boundary list is therefore actually a network, when the multiple levels of details are considered, but it can be used as a simple list at a given level of detail by traversing the links at that level only. In the course of my implementation, I found that the two most useful ways to search the boundary are at the most coarse and most fine levels of detail.

Extended Polygon Boundary Summary

The second part of the next chapter describes a reasoning system which uses the extended polygon boundary representation, and partial distance ordering (described together as EPB/PDO). The following advantages and disadvantages of the representation are a few of those noted during the implementation of that system:

Advantages:

- The ability to define a coarse boundary simplified much of the reasoning, because overall relationships are quickly lost when traversing

a boundary which consists of a chain of fine details with imprecise relationships between them. The coarse boundary description was used more often than the fine one for most operations.

- The definition of proximity as boundary elements that “might contact each other” provided explicit information useful for reasoning about object interaction.
- The partial distance ordering implicitly supports graceful degradation with inexact information, and the reasoning system based on it is naturally robust as a result.
- The doubly-linked circular list formed by boundary `left` and `right` pointers was a useful aid to boundary operations. The resulting boundary representation is similar to that used by Faltings in the `CLOCK` system [FNF87], but is more closely related to actual shape than that system, which explicitly links “first” and “last” elements of the boundary.
- An EPB/PDO representation could be constructed directly from the output of vision systems such as those described by Brady, and by MacKerras.

Disadvantages:

- Shapes described with this representation seem to be far more difficult for a person to reconstruct than with the ASSF representation. Although the representation made it easier for a computer program to reason about object interaction, it is not necessarily a good representation for people to deal with (as would be necessary in robot programming).
- Description of the relative position of objects only in terms of boundary elements made it difficult to reason about relative motion of objects. Most motion operations extrapolated from an element position to produce a derived object orientation, and then used this description for reasoning with. It may have been more appropriate to include object-scale orientation explicitly.
- Direction is represented very imprecisely, and the relative directions of boundary segments on opposite sides of an object may be completely

unknown, because of the cumulative effects of unknown angles around the boundary between them.

Overall, the extended polygon boundary representation is simpler than the ASSF representation, and it appears to offer more power for reasoning about interaction between objects in two dimensions. The main reason for this is that important factors in object interaction are explicit in the EPB/PDO representation. These factors include the representation of the boundary itself, where interaction takes place, and of the relationships between boundary elements on different objects, which can be used to identify possible interactions.

4.4 Extensions to the Two Dimensional Qualitative Representations

This chapter has presented two methods for qualitative representation of two dimensional shape: one has an emphasis on constructing two dimensional shape descriptions from hierarchical combination (through subparts) of primitive shape features, using axes to describe size, position, and orientation. The other has an emphasis on segmenting the boundaries of two dimensional shape into elements that are qualitatively homogeneous, with position described in terms of proximity between elements, and a global size ordering.

Each of these has been used to construct scene descriptions in the LISP language, and has been used as input to one of the simple planning systems described in chapter 5. The last three sections of this chapter describes possible extensions to the implemented representations which were not required for the examples, but would add both power and generality for applying a qualitative shape representation to a wider range of problems.

4.4.1 Including Order of Magnitude Information in the Distance Ordering

The partial distance ordering of size information described above can be used to make only a limited number of deductions about the scene described. It can tell us that the thickness of a sheet of paper is smaller than the depth of a file box, and that it will therefore not protrude above the top of the box if it is placed inside. If two sheets of paper are placed on top of one another, however, we can make no conclusions at all about whether or not

they fill the box. This is because we may know that one size (the box depth) is larger than another (the paper thickness), but we do not know *how much* larger it is.

An answer to the question of how many sheets of paper will fit in the box could easily be found by a system that had absolute information about the thickness of the paper and the depth of the box. Absolute information is not required, however; a person can answer this question without measuring the thickness of the paper. The qualitative knowledge that supports this reasoning is the fact that the thickness of a sheet of paper is of a different order of magnitude from the depth of the box.

Order of magnitude information can be included in the partial distance ordering by the use of **much-larger**, **much-smaller** and **nearly-equal** size relationships in addition to those for **larger** and **smaller**. Such a provision enables not only comparisons of size after addition operations as in the above example, but a complete qualitative arithmetic as discussed by Raiman, in his paper “Order of Magnitude Reasoning” [Rai86].

Order of magnitude information is easy to incorporate into the partial distance ordering, because it can simply be superimposed on the relative size information, leaving the previous ordering intact. Where such information is given, it provides extra useful data, but a scene description could be composed without any order of magnitude information. In this case, the reasoning system could continue to operate, but there would be more questions that it could not answer.

The addition of order of magnitude capabilities does not therefore affect the graceful degradation of the system when given incomplete data, but it does provide it with more power when more information is available.

4.4.2 Ordering of Angle Sizes

The geometric description of shape involves both magnitude and direction information. In the representation described above, all distances in a scene are related by their magnitudes, but angles are simply divided into four broad categories by quadrant, with the boundaries between quadrants providing three distinguished points in the angle quantity space.

This is sufficient for expressing qualitative direction of neighbouring boundary segments, and for determining overall convexity or concavity around a simple boundary. The addition of a further four qualitative regions by including **very-acute** and **very-obtuse** angles (both **concave** and **convex**) allows several trigonometric operations to be included in the repertoire of

geometric techniques for the reasoning system. Table 5.1 lists a set of rules in qualitative trigonometry which become considerably more powerful with the addition of these further qualitative angles.

Further information about angles could be represented by creating a partial ordering of angle sizes similar to the distance ordering. The requirements for angle information are rather different to those for size, however. Absolute distance values need never be expressed, as long as any required relationship between distances in the scene can be represented. The one exception to this is the contact distance (zero), which is a distinguished absolute point. There are a number of distinguished absolute points in the angle quantity space, however, because there are particular angle values which always have the same properties (there are no geometric properties that are always associated with a given absolute size).

The current system has distinguished points for qualitative description of angles at 90° , 180° , and 270° (**straight**, **convex-right** and **concave-right** angles). The **very-acute** and **very-obtuse** ranges referred to in the next chapter might also start at a distinguished point; some of the trigonometric properties assigned to **very-acute** are true of any angle less than 45° , for instance. The addition of a number of other distinguished points could also be justified by the fact that they correspond to useful trigonometric properties. These might include 30° , 60° , 110° , 150° , 210° , 240° , 300° and 330° , to name a few.

The need for a greater number of distinguished points in a partial ordering can be easily accommodated by using special absolute value nodes in the network. The **contact** list in the distance ordering is one such special node. The approach taken by Simmons in his common-sense arithmetic system [Sim86] allows any node in the network to be assigned an absolute value, and his method could be used to create a partial angle ordering, in which the nodes have an absolute value slot that can be used for normal trigonometric operations where they are necessary. This would maintain the robust nature of the qualitative representation, while providing an adjustable number of absolute distinguished points.⁴

⁴The minimum number of absolute points would be the three points that distinguish between **concave** and **convex**, and between **acute** and **obtuse**. This could provide a relatively easy transition from the current system to the angle ordering, by simply replacing the current set of distinguished angles with absolute points in the partial ordering.

4.4.3 Explicit Links to Three Dimensional Shape

The most difficult issue in applying either ASSF, EPB/PDO, or any two dimensional qualitative representation to a real robot reasoning system is the extension of the methods developed to three dimensions. The issues involved are too complex to discuss in any depth, but some ideas of initial directions to investigate include the following:

- Both generalised cones and boundary representation solid modelling techniques are based on two dimensional descriptions at a certain level. A boundary representation must include 2D faces as part of the boundary, and the generalised cone technique requires a method for describing two dimensional cross-sections.
- The majority of actions in a mechanical workspace involve simultaneous motion in no more than two dimensions. People also appear to have difficulty reasoning about motion in three independent dimensions, so limiting a robot reasoning system to two dimensions for actual motion planning may be an acceptable restriction if human-like performance is the goal. These two dimensions might be in either a cartesian or polar co-ordinate system, since simultaneous turning and linear motion in the case of screw operation is not a difficult reasoning problem for people.
- In two dimensions, there are only three types of possible contacts between object boundaries: junction–junction, segment–segment, and junction–segment. In three dimensions, there are six types of possible contact: plane–plane, plane–edge, plane–vertex, edge–edge, edge–vertex and vertex–vertex. This requires that planes be explicitly represented, and also that methods of expressing relative orientation and contact be more sophisticated. This would require a completely different approach to defining proximity (which must reflect potential contact).
- Wong and Fu describe a method for planning motion in three dimensions, operating in the space of a two-dimensional orthogonal projection [WF86].

Chapter 5

Implementations of Two Qualitative Spatial Reasoning Systems

This chapter describes two programs which perform qualitative spatial reasoning using qualitative scene information. Each program can solve a basic spatial reasoning problem. The reason for constructing these programs was to test the capabilities of qualitative spatial representations in problem solving tasks, and this chapter discusses the grounds on which the two representations in the last chapter were evaluated.

The tasks used to test these representations were described in the introduction. One task is to find out what surfaces a moving object can come into contact with when it is sliding around a number of obstacles. The other is to plan a path for a moving object through a group of stationary obstacles, and into free space. The program which solved the first of these tasks used the ASSF representation, but it could equally well have been based on EPB/PDO. The program which solved the path planning problem used the EPB/PDO representation.

The programs described were both implemented in the LISP language, and some mention is made of language related matters in this chapter. This is because the descriptions of my programs are intended to be sufficiently detailed to make another implementation possible. The level at which description is carried out, however, does not extend to lisp coding details, and it should be possible to implement a similar system in a language other than LISP by using this chapter as a reference.

5.1 A Program for Reasoning About Sliding

I proposed the “sliding problem” as an initial case study to test the power of the ASSF shape representation. The two dimensional scene in which the task is carried out includes a number of stationary obstacles, and one moving object. The task for the program is to list the contacts that might occur if the moving object were sliding around the obstacles (the only constraint on motion in sliding is that the moving object must stay in contact with at least one of the obstacles at all times).

The possible contacts which the program must find are qualitatively described in terms of the pair of features which are in contact: one on the moving object, and one on an obstacle. The moving object might be in contact with more than one obstacle at one time, so one entry in the set of possible contacts might include several of these feature pairs.

The sliding problem is not an especially difficult one when compared to human performance in spatial reasoning, or even when compared to some robot reasoning systems. It is, however, a problem which would not be trivial to solve using computational methods based on numerical geometry. The main point of interest as a problem in qualitative reasoning is that the solution must be expressed in terms of qualitative state; the definition of a “contact” implies a particular qualitative state, whereas a numeric system could provide only a range of coordinate values over which the position constraints of the problem are met.

The overall method described below for solving this problem could be implemented equally well using any qualitative shape description that includes a description of boundary features. The last part of this section discusses some technical details of my implementation using the ASSF representation, but a number of these points would apply to any representation.

The main challenge in this problem lies in the need to qualitatively describe the process of sliding, and the state of contact. Both involve interaction between objects that takes place on the boundaries of those objects. The two are closely related by the formulation of the problem, which defines the constraint on sliding motion in terms of contact, and asks for a set of contacts which can occur through sliding. It is therefore natural to conversely describe the process of sliding as a change in contact state over time.

In order to solve the problem, a program must therefore be capable of remembering more than one state of the system under analysis, and it must be capable of ordering these states with respect to the times at which

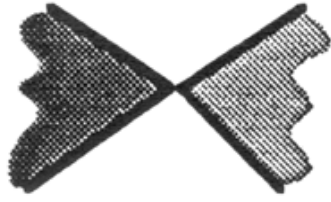


Figure 5.1: Vertex to Vertex Contact

they occur. It must also be capable of representing and analysing state transition. The following sections discuss the representation of contact state, the analysis of state transition, and finally the overall reasoning strategy followed by the program.

5.1.1 Representing Contact State

In terms of the ASSF representation, a “contact” is one item on the list of things that are in contact with a particular feature. Each item on this list refers to a second shape feature (on a different object) that is in contact with this one. The qualitative state information implicit in this representation of a contact includes the names of the two objects that are in contact, the relative size of the features which are in contact, the relative orientation of the objects which they belong to, and the qualitative shapes of the features, and their neighbouring features.

The most important of these qualities for the purposes of sliding analysis is the shape of the two features. Different sliding behaviour can be observed for each of the differently shaped feature pairs described in the following list:

1. Contact between two vertices only. This results in each vertex appearing on the contact list associated with the other. (See figure 5.1)
2. Contact between a line segment and a vertex. Each appears on the contact list of the other. (See figure 5.2)
3. Contact between two line segments of the same length, which are exactly aligned. Each line segment appears on the contact list of the



Figure 5.2: Line Segment to Vertex Contact

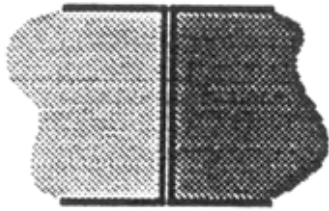


Figure 5.3: Aligned Line Segments Contact

other. In addition to this, the vertices at each end of each segment appear on the contact list associated with the vertices at each end of the other segment. (See figure 5.3)

4. Contact between two overlapping line segments. In this case, the contact list of each line segment includes the other line segment, and also the vertex at one end of the other. Each vertex also has the other line segment on its contact list. The side of the line segment on which the contacting vertex appears separates two types of overlap contact – either both vertices are clockwise from the contacting segment when the boundary of the object is being traversed, or both are anticlockwise. (See figure 5.4)
5. Contact between two line segments, where one is longer than the other, and the longer one extends past the shorter at both ends. In this case the contact list of the longer will include the shorter segment, and also the vertex at each end. The contact list of the shorter will have only

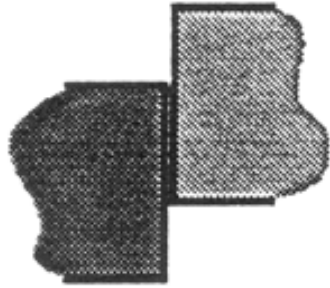


Figure 5.4: Overlapping Line Segments Contact



Figure 5.5: Large Line Segment to Small Line Segment Contact

one entry – the larger line segment. (See figure 5.5)

6. Contact between a vertex and a convex curve. Each appears on the contact list of the other. (See figure 5.6)
7. Contact between a vertex and a concave curve. Each appears on the contact list of the other. (See figure 5.7)
8. Contact between a line segment and a convex curve. Each appears on the contact list of the other. The vertices at the ends of the line segment will not be in contact with the curve. (See figure 5.8)
9. Contact between a line segment and a concave curve is not possible, but the vertices at each end of the line segment can be in contact with



Figure 5.6: Vertex to Convex Curve Contact



Figure 5.7: Vertex to Concave Curve Contact

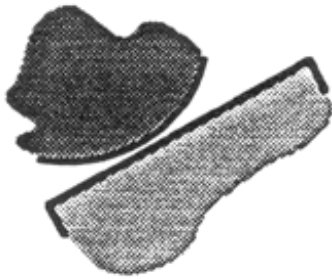


Figure 5.8: Line Segment to Convex Curve Contact



Figure 5.9: Line Segment to Concave Curve Contact



Figure 5.10: Convex Curve to Convex Curve Contact

a concave curve. (See figure 5.9)

10. Two convex curves can contact each other. (Concave curves cannot.) (See figure 5.10)
11. A concave and convex curve can contact each other only if the angle of the convex curve is smaller than the concave curve. (See figure 5.11)

In addition to the actual shape of two features in contact, it is apparent from the above list that the shape and contact status of neighbouring features is also relevant in distinguishing between qualitative contact states. The shape of neighbouring features becomes more important, however, when considering the ways in which the contact state can change as the objects slide past each other.



Figure 5.11: Concave Curve to Convex Curve Contact

5.1.2 Contact State Transitions

All possible sliding motions involve changes of state between the above categories of contact, and this is represented by corresponding changes in the contact list for the moving object, and in those of the obstacles. The change in state which occurs after any given sliding motion depends on the shape of the features involved, and on the direction of the slide.

The shape of features which are neighbours to the current contact site is static information which can be obtained directly from the qualitative scene description. The direction of sliding motion is dynamic information, which must be determined for each change of state. Direction of sliding is conveniently described for a single moving object as being either clockwise or anticlockwise with respect to the boundary of the moving object (for multiple moving objects, direction would have to be stated relative to one of them).

The distinction between the clockwise and anticlockwise sliding directions in motion description must be supported by an annotation of shape description that establishes which neighbours of a feature lie in a clockwise or anticlockwise direction (this can be stated independently relative to the boundary of each object). If this were not done, the reasoning program would not be able to determine the “handedness” of the object descriptions, and would not be able to distinguish between mirror copies of objects.

A substantial amount of effort in the implementation of this sliding analysis program involved providing functions for accessing the shape representation in terms of the two directions, and describing both contacts and object boundaries in these terms. A typical problem in doing this was that the list of features which comprised a subpart or object was described as a simple

list, whereas the clockwise ordering of neighbouring features had to be circular. The necessity for use of circular lists is mentioned again briefly in the discussion of implementation.

Having established the present state, the neighbouring features to the current contact position, and the direction in which motion toward the neighbouring features is taking place, it is possible to identify the next qualitative change in contact state that will occur.

The list of possible state transitions is much less than the square of the number of states, because very few states can possibly follow each unique contact type in a physically possible object description. In particular, any contact between non-vertex combinations must be followed by a contact that involves vertices. Possible transitions can be categorised in terms of their behaviour around vertices. The program described here distinguishes between three important types of sliding state around a vertex. I describe these as “aligned” contact, “sliding over” a vertex, and “falling off” a vertex.

- All contact states can be classified as either **aligned** or **non-aligned**. Aligned contacts are states in which any motion at all will alter the state, because there are vertices in contact with other vertices. There will always be an **aligned** contact state in between two **non-aligned** states, except when concave and convex combinations of features are in contact. At this point, the moving object is about to start (or has just finished) sliding over a vertex. These aligned states therefore mark a transition between sliding over edges, and sliding over vertices.
- It is only possible to **slide-over** convex vertices. Concave vertices require either that the moving object rotate, or that the aligned vertex on the moving object continue to slide with a vertex to edge contact over the following edge. There is therefore a second kind of motion that can occur around an aligned state: a combination of **slide-to**, and **slide-from**.
- If a line segment or curve is sliding over a convex vertex, and there is another convex vertex at the end of the moving segment, it is possible for the moving object to **fall-off** the obstacle that it is sliding around. The actual behaviour will depend on the free space motion characteristics of the scene space (in particular whether there is any gravity, or similar field). Possible behaviours which satisfy the sliding constraint are that the moving object rotates about the fall-off point, or slides-from the aligned vertex.

There are therefore three basic types of **aligned** to **non-aligned** transition: the **slide-over** motion, the **slide-to/slide-from** motion, and the **fall-off** motion. These transition categories, when expanded to include the relevant cases from the list of possible contact states, can describe the motion of a single moving object over the boundary of a single obstacle, but they must be extended to describe an object moving around a number of obstacles. This extension is not too complex, because two obstacles in contact (or even near each other) form a new feature that behaves like one of the normal features. The extensions that would be necessary identify the equivalent feature, and allow the concept of “neighbouring” features to include features on other objects which are candidates for the next contact.

Transitions between contacts with features on different objects are therefore of the same types as transitions between features on the same object. The main distinction is between the cases of sliding over an aligned boundary between objects, and an overlapping boundary. The simplest case is the aligned boundary, in which there is an intermediate state where the moving object is in contact with both objects.

There are several cases of overlapping obstacle contacts: there are two directions from which to approach an overlap – from the high side, the moving object will “fall off” onto the second obstacle. From the low side, the moving object will come into contact with the hanging part of the overlap. In either case, the behaviour also depends on the curvature of neighbouring shape features.

The level of analysis presented so far does not deal with one important case: contact between a concave and convex curve, or even between the areas surrounding a concave and convex vertex. If the angles of the curves or vertices include right angles, or if one is acute and the other obtuse, the resulting contact states can be determined. If both are acute or both obtuse, it is impossible to determine the contact that will occur, because the relative sizes of the angles are unknown.

A second case which is difficult to analyse is sliding across the gap between two obstacles which are not touching each other. This would require direct comparison between the size of the gap between the obstacles, and the size of the moving object. The other state transitions I have described can all be qualitatively analysed without size or angle magnitude comparison, and it was this case which made clear the deficiencies of the ASSF representation with respect to direct comparison of feature sizes. This is discussed further with other implementation issues.

5.1.3 Envisionment of Motion through Contact States

In any qualitative envisionment, a possible change in state of the system being modelled must be reflected in the representation, so that future state changes can be determined. The complete dynamic state of the sliding object system can be described as the set of all contacts in the scene. As mentioned above (and presented in more detail in the implementation section), contact is described relative to each feature, as a list of things that are in contact with that feature.

The global contact state for the whole scene is the set of all such contact lists. Each contact is mentioned twice in this set, once in the list of each feature that participates in the contact. Updating the contact set to reflect a change of state involves adding new contacts to the set, and removing all reference to contacts that have been separated.

Adding new contacts includes the addition of new contact lists to the set if a feature has no previous contacts, and determining whether the contact should be at the clockwise or anticlockwise end of the contact list if the feature does have previous contacts. Where a contact is removed, it must be removed from the contact lists of each feature to maintain consistency in the contact set. All of these functions, although they are necessary to maintain consistency in the contact set, are analagous to principles that humans automatically apply when reasoning about relative motion of moving and stationary objects.

A higher level of reasoning in the program makes use of this contact set information to search for possible new contact configurations. The search tree is presently only a binary tree, because at any point the moving object may only move clockwise or anticlockwise. If the program dealt with obstacles which had gaps between them, this search tree might expand, because of the multiple possibilities in moving between objects.

The search level of the sliding system collates a “history” of possible contact sets. It also determines from the contact history when it has exhausted the possible contact configurations (this occurs when a system state is exactly repeated, and the program has searched from that state in both clockwise and anticlockwise directions).

The solution required to the original sliding problem is an envisionment of all contact sets that can occur as the moving object slides. This contact history can therefore directly provide the final solution.

As pointed out in chapter 3, planning tasks differ from other qualitative reasoning domains, in that the “envisionment” is actually the effect of ac-

tions planned by the reasoning program, whereas “function from structure” problems usually determine the behaviour of a system that changes state by itself. The envisionment described here can therefore be considered by a planning program to be a history of the states that would have occurred if a robot had actually carried out previous planning stages. The formal relationship between history and envisionment in qualitative physics reasoning is discussed in some detail by Forbus [For87]. The attitude I have taken in this implementation is however quite informal.

5.1.4 The ASSF Implementation

The program implemented to solve the sliding problem made use of many of the facilities provided in the ASSF proposal, but did not include some aspects. In particular, it omitted the description of axis relative object positions. The parts of the proposed representation which were important for solving this problem were the qualitative description of feature shape, including vertex angles, the relative position of features on an object by “neighbour” relationships, and the contact list associated with each feature.

The ASSF representation was implemented with extensive use of data types that support multiple attributes of different types. My implementation used the LISP “defstruct”, but any method of relating data items of different types to an single reference could have been used. The shape elements described in this way included basic features (vertices, lines and curves), orientation and position, and functional groupings of features for generic part or object shape types.

The attributes of a feature vary between vertices, lines and curves. The attributes of a line feature include length, orientation, and position relative to the local object. The attributes of a vertex include its angle, the orientation of the angle bisector, and whether it is concave or convex. The attributes of a curve include orientation of the chord axis, length of the chord, position of the chord centre, bulge of the curve, and whether it is concave or convex. The “wiggle” feature type was not included in this implementation, since sliding over wiggles involves a different type of reasoning problem.

Position and orientation descriptions are also defined by defstructs, since position and orientation have several components. These components include major and minor axis positions, and local and global orientation. Although the facility for describing global orientation was included in the representation, it was not used for the solution of the sliding problem.

The use of groupings of features as generic shape types allowed common two dimensional shapes to be defined once, so that any given shape in a scene is simply an instance of the generic type. As an example, a generic square is defined as composed of four edge features, and four vertex features. There are certain attributes of each feature which can be defined as being true for all squares. The definition of the square includes information about where these features are on the boundary of the square, and how they are related to each other. The relationships described include sets of lines which are parallel and pairs of lines which are perpendicular to each other, as well as the requirement for right-angled vertices in a square.

Any reference to one part of a particular two dimensional shape in the scene is made using terms established in the generic shape definition. One side of a particular square, for example, might referred to as the second line segment in the third instance of generic shape number one.

Scene and Problem Representation

When establishing possible sliding motion, the most important information about the scene is the description of what features or objects are in contact with other features or objects. These contacts were represented using “contact lists”. Whenever one shape feature comes into contact with another, a contact list is created for each feature. The contact list is linked to the feature by the name of that feature at the head of the list. The rest of the list consists of all features which are in contact with the feature at the head. Any further contacts that occur can be added to this list.

The contact lists are ordered in a clockwise direction (relative to the object on which the head feature is found). This implicit direction in the contact list ordering is used as the basis for distinguishing between clockwise and anticlockwise sliding, since the system must be able to tell what relative direction successive slides occur in.

All contacts between objects in a scene are collected in the “contact set”, which is the set of all contact lists. The contact set contains two entries for every contact; one entry is found on the contact list of each of the two features in contact. It is necessary to maintain consistency in the contact set by removing entries from the relevant contact lists when a contact is broken, and removing whole lists when the feature the list refers to is no longer in contact with anything.

For ease in formulating the problem, one object in a scene is represented directly as the `moving-object`, while all others are called `obstacle-1`,

`obstacle-2` etc. The analysis of sliding therefore commences with the contact lists for each feature of `moving-object`.

In order to represent motion around the boundary of a single obstacle, the features on the boundary must be ordered. This is also achieved by using an implicit clockwise ordering in the list of features that are in a given subpart or object. Use of a linear list for this purpose created some problems, in that the clockwise ordering of features on the boundary is circular. It is possible to create circular lists in LISP, but they are difficult to operate on – the reasoning system therefore had to allow for a “wrap-around” from the last item in a list of features to the first, and vice-versa in the anticlockwise direction.

Although size comparison between features on different objects would have been useful, this system was not able to easily perform such comparison. This resulted from the fact that feature sizes are expressed by the relationship between the length of the feature axis, and the length of the main object or subpart axis. The relative sizes of whole objects were expressed in a similar fashion. Comparison of two features therefore required following a chain of qualitative size comparisons through two objects – by the end of the chain, it was seldom possible to make any clear conclusion of size relative to the feature at the start of the chain.

As described above, the sliding system searches exhaustively for all possible contact states. This exhaustive search terminates when a previous contact state is repeated (when the moving object has circumnavigated all obstacles). Detection of a repeated contact state is not trivial, because there is no canonical ordering of the contact set (even though all contact lists and object feature lists are guaranteed to be clockwise-ordered, they can start and end at different points). It was therefore necessary to construct a function for comparing overall scene configurations after each sliding motion.

5.1.5 Summary of Sliding Issues

The essence of the sliding problem, both in the constraints specified, and in the required solution, is that it deals with contacts between different objects. The importance of contact to this basic problem immediately demonstrated a deficiency in the ASSF representation as presented in chapter 4. The ASSF representation described relative position of objects in terms of the object axes, and this meant that any contact between the objects had to be inferred from relative orientation and magnitude of objects, subparts, features, and axes.

The global contact set therefore provided a useful way of operating directly with necessary state information, rather than having to derive it from physical shape. The contact set was not ideal as a state description, especially because invariant information (contact between obstacles) was mixed up with state information (contact between the moving object and obstacles). It did, however, provide the basis for a reasoning system that operated with ordered states – the contact history.

The contact history was a clear and natural way of representing motion. All changes of state were recorded, and the use of a qualitatively significant criterion for state change meant that most of the contact history information was significant to the problem analysis. This is not always the case in such simple history mechanisms, and it shows an advantage of the qualitative representation, in that operations performed using qualitative data produce more qualitatively interesting results.

The boundary list maintained for each object in the ASSF representation was used rather more heavily than anticipated – in particular, it was essential for the boundary list to have a strict clockwise ordering. This was not my original intention for the ASSF system – on the contrary, those features on the boundary of a shape were to be recognised as such by their position attributes, in particular the fact that the feature was located at the full extent of an object axis. For the sliding problem, however, all features that can be involved in contact are on the boundary, and the fact that this list was created by assuming a traversal of the boundary in a clockwise direction provided a far more useful reference point than any data relating to shape axes.

The importance of the boundary meant that a number of special LISP functions had to be created to make the boundary easier to work with. Examples of these are the “wrap-around” list operations to make the simple linear list appear to be a continuous ring of clockwise and anticlockwise neighbour relationships.

A more major issue was that the methods used to solve the sliding problem make use only of a relatively small subset of the facilities provided in the proposed full ASSF shape representation. As an initial investigation, it did not need most of the axially specified facilities, but attempts to extend the sliding approach to include free-space motion soon showed that the representation techniques for size, position and orientation were too clumsy for the task of reasoning about complex and/or underconstrained interaction between objects, even though it provided a very full description of shape for individual objects.

It became clear in the course of this implementation that since interaction between objects (whether or not it involves sliding) occurs exclusively between the boundaries of the objects, the shape representation used should therefore concentrate on providing an adequate boundary representation. It was this concern that led to the development of the EPB/PDO representation, because ASSF, while adequately meeting my initial goal of a completely qualitative shape description method, failed to fully meet the requirements of reasoning about motion and object interaction.

5.2 Reasoning About Path-Planning with EPB/PDO

5.2.1 The Path-Planning Problem

Path planning problems are a far more common concern in robotics than the simple sliding problem presented in the previous section. The typical problem formulation in path planning is the “(piano) mover’s problem”, or “findpath problem”. This problem, like my sliding problem, involves a number of obstacles and a single moving object. The difference between the mover’s problem and the sliding problem is that in the mover’s problem, the goal is to find a path through the obstacles while avoiding touching any of them (rather than finding what can happen when maintaining contact with them).¹

The obstacles in my examples are all right angled polygons, with a mixture of concave and convex vertices, and with multiple levels of detail on the boundary. This is also true of the moving object. Most of the implemented code allows for angles which are not right angles, and for curved edges, but examples with these properties were not tested, for reasons explained later.

The reasoning performed in solving this problem was intended to complement that performed in the sliding problem. It involves motion of the moving object through free space, and discovery of non-contact paths between obstacles. The goal state is a position for the moving object which is on the other side of the obstacles initially surrounding it, and provides unbounded movement away from those obstacles. The solution to the problem also includes a suggested path which will enable the moving object to reach that goal state.

The remainder of this section first discusses the implementation of the

¹See [LP83] for a typical geometric algorithm which finds solutions to the mover’s problem.

extended polygon boundary/partial distance ordering representation, and then describes the reasoning strategies used to solve path planning problems. The overall stages in solving a problem are discussed first, and the method of solution for each stage is then described.

5.2.2 EPB/PDO Implementation²

The most basic level at which an object boundary is represented is an unordered set of boundary elements listed as a “property”³ of the object. Objects have no other properties for the purpose of this system, but in a more extensive reasoning system, the **object** may provide an interface between a higher level of reasoning and the path planning mechanism.

The set of boundary elements includes both **segments** and **junctions**, and these are distinguished by a **type** property, which identifies them as one of the two types. The ordering of elements around the boundary is established using “neighbourhood” properties on each boundary element. The **left** property points to the boundary element on the left (anticlockwise) of this one, and the **right** property points to that on the right. The **left** and **right** pointers can be used to treat the boundary as a doubly linked circular list.⁴

The **left** and **right** neighbourhood properties are not actually simple pointers, but lists, where the list is ordered by detail resolution; the first element of the list is the coarsest level of boundary detail, and the last is the finest level of detail. The neighbourhood description is therefore used to implement the web of boundary relationships at multiple levels of detail that was proposed in chapter 4.

Boundary elements of both the **segment** and **junction** type have a **shape** property. The **shape** of a segment may be either **line**, **curve**, or **wiggle**. **Curve** and **wiggle** shape properties also include **angle** and **bulge** (for a

²Appendix A contains a complete LISP description of a scene using EPB/PDO. The reader may wish to refer to this appendix for examples and further detail than is given in this discussion.

³Where the ASSF implementation made extensive use of the LISP “defstruct”, the EPB/PDO system was implemented using “properties”. A LISP atom can have any number of properties, and new properties be defined dynamically (defstructs must be explicitly defined by the programmer). Use of properties is more portable between different LISPs, and the ability to define them dynamically seems to be more consistent with the general style of LISP programming.

⁴Although a logical circular list is formed by these pointers, no list exists in the LISP sense – it cannot be operated on by the LISP list operations.

curve), or wiggleness and bulge. The shape property of a junction lists its **angle** and **flex**, which may be either **convex** or **concave**.⁵

Proximity of two boundary elements (or contact between them, since contact is a special case of proximity) is represented as a property of each element. This is in contrast to the ASSF implementation, where relationships are described globally, and the relationships associated with any particular element must be found using a global search.

A proximity (or contact) relationship between two elements is represented by an independent pair of pointers to the two elements. Each element has a property which lists all such proximity pairs that refer to it. There are in fact two of these lists: the **internal** proximity property refers to proximities between this element and other elements on the boundary of the same object. Internal proximities are invariant for rigid objects. The **external** proximity property refers to proximities between this element and elements on different objects. These proximities can change when objects move.

All segments also have a size property. This points to a global “equidistance” list – part of the partial distance ordering – which lists all distances in the scene that are the same size. The equidistance list specified by a segment’s **equidistant** property will have at least one entry – the segment itself.

The partial distance ordering, which relates the magnitude of all segment sizes and proximities in the scene, is composed of a number of equidistance lists. Each list specifies a set of equal distances – segments whose size is equal to this distance, and proximities between elements which are separated by this distance. The partial ordering is established with two properties of each equidistance list – the **larger** and **smaller** properties. Each of these is a pointer to another equidistance list. One special equidistance list is the **contact** list. It has no **smaller** property, and its value is the list of all boundary element proximities where the elements are in contact. In a consistent distance ordering there will also be at least one equidistance list which has no **larger** property.

There are five types of atom which may be pointed to by an entry on an equidistance list. Segment size is one of these, **internal**, **external** or **overlap** proximity pairs are another three, and the last is the **synthetic** distance, which is the result of some geometric construction carried out during problem solving (this is discussed further in section 5.2.5).

⁵LISP implementation note – shape properties are implemented using **assoc** lists with the keys **angle**, **bulge**, **wiggleness**, and **flex**.

A portion of a scene representation, including the relationship between objects, segments, junctions, proximity, and the partial distance ordering, is illustrated in figure 5.12. This figure (which is a portion of a simple scene in which a tripod is to move through a doorway) shows the representation of an object boundary using `left` and `right` neighbourhood pointers between segments and junctions. It also shows the arrangement of proximity pairs relating boundary elements on different objects, and the comparison of segment size to proximity in the partial distance ordering. It does not include object shape descriptions (multiple levels of boundary detail, or the shape of individual boundary elements).

5.2.3 Stages in Path Planning

The overall strategy of the path planning system includes three general stages (with a certain amount of interaction between the three). These three stages are:

1. Find a gap between the obstacles surrounding the moving object in its current position.
2. Check that the gap is big enough for the moving object to fit through.
3. Specify directions of motion which take the moving object to the opening of the gap, and then through it.

These three stages appear relatively independent, but they affect each other in the following ways:

- Where there is more than one possible gap, each candidate must be evaluated in terms of its suitability for later planning stages. Information about fit and motion must therefore be available to the gap finder.
- The procedure for establishing the presence of a gap also establishes the narrowest extent of the gap in terms of the partial distance ordering. This information will later be necessary when checking fit, so must be passed on from the gap finding stage to the fit checking stage.
- In addition to the narrowest extent of the gap, an entry point must be recorded, relative to which motion into the gap can be specified. The gap finding stage records this information for use in the motion specification stage.

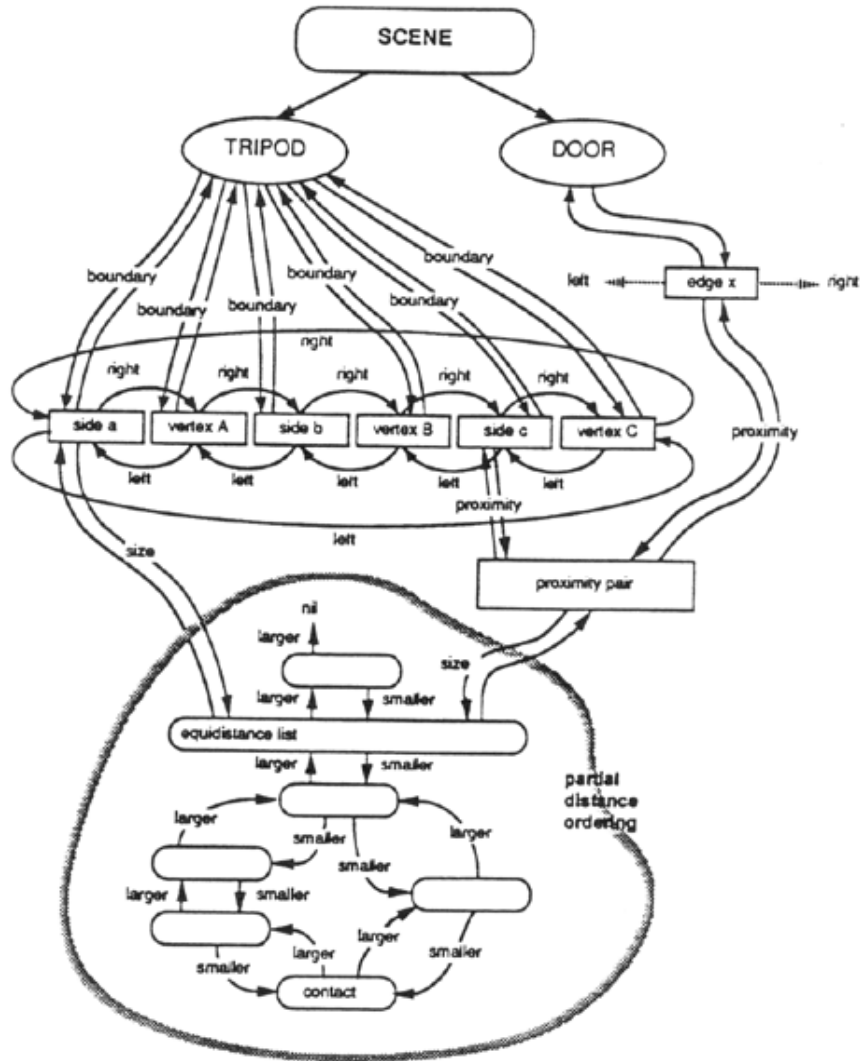


Figure 5.12: Implementation of the EPB/PDO Representation

- Fit depends on the orientation of the moving object, which is determined in part by the direction of motion when approaching the gap. This means that some information about direction of motion must be available in the fit-checking stage, before motion is planned.

These factors make it impossible for the “stages” to be carried out independently, but it is still possible to separate the types of reasoning that must be carried out to solve each part of the problem.

The path planning strategy becomes more complex if more complex fields of obstacles are to be dealt with (where the moving object has to move in turn through more than one gap). The implemented system does not deal with this case, but it is discussed along with other possible extensions in the final section of this chapter.

5.2.4 Gap Finding

The first task carried out by the gap finding part of the path planning system is to establish whether path finding is necessary at all. If there are any directions in which the moving object is not facing any obstacles, then it can easily move into free space, by advancing in that direction. This initial test is carried out so that the gap finding module can be easily integrated into a more complex planning system. It would normally be used, for instance, to identify the goal state of an obstacle field negotiation problem.

If there are no sides of the moving object which are free of obstacles, the obstacles on every side are examined to find out whether there are gaps between each obstacle and its neighbour.

Note that the term “side” when applied to the moving object means here a segment on the coarsest boundary level. Initial motion planning takes place at this coarse level, because details of the object shape only become significant during actual motion. A “gap” is a path around the boundary of an obstacle which starts at an obstructing boundary element, and ends at a boundary element that meets the “free space” criterion. The obstacle boundary between these two elements must not contact any other obstacle.

The search procedure used to find gaps is evident from this definition. For every obstacle, the obstacle boundary is searched in both directions, starting from a boundary element which is in proximity to the moving object. The search is terminated when a portion of the boundary is in contact with another obstacle, when a free-space element is encountered, or when the searches from each direction meet.

Along the search path, a record is kept of the narrowest extent of the gap. This is the smallest distance from an element along the gap to any other obstacle, as found in the partial distance ordering. At any point, the search for the nearest neighbour of a boundary element must exclude the moving object itself (since proximity to the moving object will be recorded in the partial distance ordering in the same way as proximity between obstacles).

5.2.5 Checking for Fit

The results of the gap finding search include a value in the partial distance ordering which describes the narrowest extent of each gap. When checking whether the moving object will fit through a gap, this narrowest extent must be compared to the width of the moving object.

The “width” of the moving object depends on its orientation as it moves through the gap, and it is necessary to find an expression for the effective width of the moving object when moving in a given direction. This is done by finding the extremities of the object at right angles to the direction of motion, and then comparing the distance between these extremities to the gap size.⁶

The current method for finding object extremities conducts a search from a given boundary element, evaluating each element encountered to find out how far it will protrude given the specified direction of motion. Candidates for extremities are placed on a candidate list, so that they can be evaluated further at a later stage. As soon as the search procedure has accumulated enough of these candidates, another set of rules is used for pairing extremities which are likely to be furthest apart. Single candidates and paired candidates are pruned from the candidate list if other candidates obviously protrude further.

Once the boundary search is complete, and a pruned set of extremity candidates is found, qualitative expressions for the distance between the extremities are derived. These expressions are related to the partial distance ordering, and can use simple rules in “qualitative trigonometry”, together with **internal** proximity information about the boundary elements of the moving object.

The system aims to find an expression for the maximum possible value of this distance between extremities. This is most easily done using the partial distance ordering, where an item of the **synthetic** distance type is used as a

⁶Note that this technique is only applicable to translation – further facilities must be provided to analyse simultaneous translation and rotation.

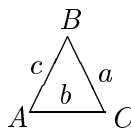
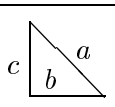
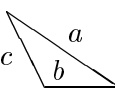
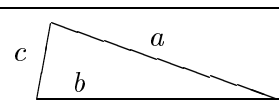
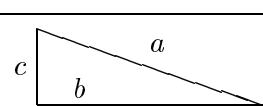
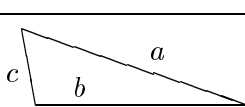
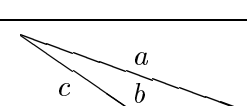
place-holder in equidistance lists for the unknown distance (the **synthetic** type distinguishes it from measured distances). In most cases, an expression for this **synthetic** object extremity can be derived either from the size of boundary segments which are parallel to the unknown extremity, or in terms of **internal** proximity.

The width found in this way may simply be equal to a known distance, or it may be derived using operations in qualitative arithmetic. Typical functions include simple binary addition and subtraction of distances in the partial distance ordering.

Where it is not possible to directly assign a value to the extremity, this is usually because there is no parallel segment, or internal proximity between perpendicular segments, that can be used to find an appropriate expression. In these cases an approximation to the value can be found using “qualitative trigonometry”. The qualitative trigonometry rules listed in table 5.1 can be used to find expressions for unknown distances across the interior of the object in terms of the size of boundary segments, or **internal** proximities. Such information is always far less constrained than the measured values explicitly entered in the partial distance ordering, however, and is not often used as the basis for a final decision on gap fit.

The last step in establishing effective width in motion is to compare the remaining entries on the extremity candidate list, and find which are the largest possible extremities for the object. In future, some further pruning of the blackboard should be performed before this stage, in order to find and remove impossibly large extremities. Such pruning would operate by comparing the extremity candidates to the proximity of obstacles in the context of the moving object, and checking to ensure the effective width is not larger than the distance between surrounding obstacles.

The final qualitative expression for width of the moving object is compared to the size of the narrowest extent of the gap, and one of three judgements can be made from the partial distance ordering: either the moving object definitely will fit through the gap, it definitely will not fit, or fit has not been established. This last case enables further investigation to be performed when other methods are available. These methods may involve the application of more rigorous geometric reasoning, or physical measurement performed by a robot. This system does not at present regard such gaps as candidates, and simply continues to search for a known “definite fit” gap.

Qualitative Properties of Triangles				
Triangle	Angles			Properties
	<i>A</i>	<i>B</i>	<i>C</i>	
	acute	acute	acute	$a < b + c^*$ $b < a + c^*$ $c < a + b^*$
	right	acute	acute	$a > c$ $a > b$
	obtuse	acute	acute	$a > c$ $a > b$
	acute	acute	very-acute	$a \simeq b$ $c \ll a$ $c \ll b$
	right	acute	very-acute	$a \simeq b$ $c \ll a$ $c \ll b$ $a > b$
	obtuse	acute	very-acute	$c \ll a$ $c \ll b$ $a > b$
	obtuse	very-acute	very-acute	$a > b$ $a > c$

* Note that these properties hold for all triangles.

Table 5.1: A Qualitative “Trig Table”.

5.2.6 Specifying Motion

In specifying motion toward a gap, the system must initially aim to position the moving object at a point from which it can negotiate the first obstacle to a straight path through the gap. This point is usually next to the junction at the end of an obstructing segment. Later motions can then be expressed in terms of further obstruction-passing points. These sub-goals for motion past obstacles are called “via points”. The currently implemented system only considers via-points which can be reached using straight line motion, but a more general system could include motion along curved lines, and also include information regarding the required rotational orientation of the moving object.

The system locates via points by finding what obstacle is on the near side of a suitable gap, and then searching for the first junction at the entrance to the gap. The via point is specified as a pair of boundary elements: this junction, and the nearest boundary element on the other side of the gap.

The proposed motion is specified as the direction from one boundary element on the moving object to one of the via point elements. This direction is then translated into a direction of motion for each boundary element of the moving object, so that any further obstructions to motion may be detected.

Directions of motion are also evaluated as to how “vague” they are. This is discussed further in the next section; it provides some robustness by allowing the system to evaluate the quality of its directional information, and employ precise directions rather than vague ones where possible.

Motion specification is currently only translational – if rotational motion was to be accommodated, substantial changes would be required. The simplest way to extend the translational planning system would be to use sequential combinations of rotational and translational motion. Reasoning about simultaneous rotation and translation in two dimensions would require the use of a 3-dimensional configuration space, and new planning strategies and directional reasoning, in addition to new methods for representing gap size, finding moving object extremities, and specifying via points.

5.2.7 Directional Reasoning

Reasoning about object motion requires the ability to represent direction. There is no explicit directional information in the original scene description, but it can be inferred from information such as the angle of curves and junctions, or from “parallel proximity”; a segment which is in proximity to

another segment, with junctions at each end which are both equidistant from that segment, is known to be parallel to it. The representation of direction requires that the reasoning system be able to relate vertex angles or parallel lines to directions of motion.

Any direction reference is specified relative to one boundary element on a one object (i.e. there is no global direction such as “North”). Directions can be specified as **forward**, **back**, **left** or **right** relative to the named boundary element, where the named direction refers to a possible motion of the object.

For directional reasoning purposes, a full set of relationships must be established between these motion directions, and vertex angles. This is achieved using functions that apply rules such as “The segment on the left side of an acute, convex junction faces backward and to the left with respect to the segment on the right side”⁷. Other directional relationships include the effects of reflection, and addition or subtraction of angles. These are all defined by a set of direction functions.

The directional reasoning part of the system distinguishes between directions that are precisely known, and directions that are “vague”. Precisely known directions are based on right angles, and represent exact angular values. **Forward**, **left**, **back**, and **right** are all precise directions. Between these directions are four quadrants (corresponding to acute and obtuse angles), within which the angle value could be anywhere within the 90 degree range. The “vague” directions describing these quadrants are “forward and to the left” (**fl**), “back and to the right” (**br**), “forward and to the right” (**fr**), and “back and to the left” (**bl**). This organisation is obviously a result of the way the angle quantity space is structured in this implementation, but the distinct nature of precise directions applies to some values in any quantity space.

It is possible to perform some operations with vague directions – the reflection of a vague direction is known (it is another vague direction), as is the value of that direction plus or minus 90 degrees (also a vague direction). No operation performed on a vague direction can return a precise direction, however, and this limits the utility of vague directional information. In addition, an **acute** or **obtuse** angle added to another vague direction results in no certain direction at all, because the result may be in either of two quadrants, or it may be a right angle.

⁷These “rules” are actually `assoc` lists which are interpreted to relate all possible combinations of angles to directions

The reasoning system includes functions to discriminate between the vague and precise directions where necessary, so that directional information can be maintained that is as precise as possible given the input data. Most functions of the system will continue to operate with vague information for as long as possible, but will return an “unknown direction” token (the unknown direction is represented by the atom `'?` in lisp) when information becomes too imprecise. The atom `'?` effectively acts as a ninth qualitative direction, which is accepted as a legal input by directional reasoning functions, but usually results in the propagation of unknown directions throughout the system during geometric reasoning tasks.

The incorporation of “vague” and “unknown” directions makes the current system robust enough to accept scene data involving imprecise qualitative angle descriptions. This directional reasoning is, however, less powerful than the partial distance ordering, and in most cases it was vague angle information that resulted in the breakdown of geometric reasoning tasks, rather than lack of precision in size information.

5.2.8 Summary of Path-Planning Issues

The EPB/PDO representation, as implemented for this system, provides a number of advantages in reasoning about motion. An important advantage is the way in which relationships between individual boundary elements are explicit properties of the elements themselves. This fact removes the need to search for information in global sets of relationships. Since most reasoning about motion involves reasoning about interaction between objects, and interaction between objects involves relationships between their boundaries, the reasoning system can be greatly simplified by making this information readily accessible.

Relative size information has been made more explicit in the EPB/PDO representation than in the ASSF representation by using the partial distance ordering. This extra information makes path planning easier, but accentuates problems caused by the lack of directional information. As stated above, the reason for failure of reasoning goals in this system is nearly always lack of precise directional information. This occurs whenever angles other than right angles are encountered – several non-right angles in a reasoning chain result in a completely unknown overall angle.

This poor directional information is allowed for by the distinction between vague and precise angles. This distinction allows the system to perform well when large numbers of right angles are in the scene, and also allows

it to recognise where problems will result when there are not many right angles. The result is a reasonably robust system for using scanty directional information, but overall performance is limited by the lack of power of the simple four quadrant description of qualitative direction and angles.

In finding the rules for “qualitative trigonometry”, it was clear that very few rules could be stated on the basis of **right**, **acute**, and **obtuse** only. The number of rules was more than doubled by the addition of two new qualitative angle ranges – **very-acute** and **very-obtuse**. This would continue to be true as new ranges were added, until in the extreme case the qualitative trigonometry table had as many entries as a numeric table. Although this would increase the power of the reasoning system, it is contrary to the objectives of this study, and the development of an alternative approach to qualitative expression of direction (perhaps based on a “partial angle ordering”, or on order of magnitude reasoning) would be more valuable.

The use of multiple levels of detail on the object boundary did reduce reliance on precise directional information. Complex pieces of boundary with many “vague” angles were often represented at a coarser level of detail with fewer angles, and this enabled the system to perform coarse path planning more confidently. In addition to this, the lack of allowance for curved boundary segments in my implementation was of little importance for coarse path planning, because most curves can be represented at a coarse level of detail by two straight segments with a junction between them, where the angle of the junction approximates the curve. The places where this is not practical include calculation of extremities, which must include the bulge of the curve rather than the position of the extra junction, and reasoning about rotation involving curves.

The implementation of this system has shown that it is possible, using qualitative reasoning techniques, to solve practical path planning problems. The natural robustness available to a qualitative system is evident in the application of the partial distance ordering, and of relative angle precision. Both in angle and magnitude reasoning, the system is able to select from different data and geometric constructions to find the most precise data available. However imprecise that data is, the system can continue with its reasoning process, until it recognises that the data is now insufficient. In this case, it is still able to act on the lack of data, and ask a robot or an operator to gather more data.

5.3 Future Enhancements

The system implemented still includes only the features required to solve the path planning problem for very simple cases, and does not include the complete features of the EPB/PDO representation as described in the previous chapter (most notably, it lacks any reasoning about curves). In addition, the methods used for checking fit through a gap need to be completed and extended. Nevertheless, the implementation carried out has demonstrated the major advantages and disadvantages of this approach, and important enhancements other than the simple completion of the originally proposed system can be identified. Three directions that I see as important initial steps for future work are described in this section.

5.3.1 Integration of two approaches

There are a number of sub-problems within the path planning problem which could be solved by the methods used for the sliding problem. For instance, the safest route through a narrow gap is to slide along one side of the gap.

Although the sliding system was implemented before I developed the EPB/PDO representation, the methods used would be easily adapted to this representation. In fact, an implementation using EPB/PDO would be simpler than the one using ASSF. If this was done, it would not be difficult to integrate the two planning methods into a system that could choose between different motion strategies during path planning.

The main characteristics of these two strategies are boundary following for the sliding system, and straight line motion between a series of via points (such as the narrowest point in a gap, and the entrance to a gap) in the path planning system. A framework which could direct either of these strategies could also include further strategies, such as rotation, wiggling into place, or other motion strategies regularly used by people. The addition of new motion strategies to an overall task planning structure provides an interesting basis for comparison to human acquisition of manipulation skills, as shown in Sussman's HACKER system [Sus75].

5.3.2 Searching for Complex Paths

The path planning system described above has the goal of reaching a state in which the moving object is in "free space". It assumes that there will only be one layer of obstacles between the current location of the moving

object, and free space on the other side of the obstacles. In a more complex scene, where the obstacles compose a “maze” which the moving object must traverse, a high level strategy is necessary to oversee the planning of paths between layers of obstacles.

The basic methods used by the present path planning system could be retained in this case; once it is possible to plan a path between two objects, it is possible in general to plan a path through many, taking them two at a time (access to a via point between two obstacles might, however, be restricted by a third obstacle).

Complex path planning of this type could be carried out with the aid of a supervisory system, which would be informed of all possible via points between obstacles, and would recommend the order in which to approach them. The supervisor would make use of some search strategy to organise these decisions. A “hill climbing” search would be sufficient, if a satisfactory function was provided for evaluating how close the moving object was to leaving the maze. This function would be used to replace the “free space” goal of the current system. If the supervisory search strategy involved backtracking, it would be necessary to extend the spatial reasoning system, providing facilities for retracting state changes in the scene representation that had resulted from previous envisionments.

5.3.3 Unfastening Problems

The initial goal of this research, as mentioned in the introduction to the thesis, was to find a technique for reasoning about unfastening. All unfastening problems can be described at a basic level as the removal of a part in an assembly into a position where its motion is unconstrained. The sliding configuration and “free space” goals used for the above qualitative reasoning systems were chosen as first steps in solving unfastening problems.

The main distinction between unfastening and the kind of planning problem discussed above is that fasteners normally involve a locking mechanism which is interlocked with the object fastened. Real world fasteners usually rely also on distortion of the fastener or fastened object. The stress resulting from this distortion, combined with friction, prevents the fastener or fastened object from moving.

Reasoning about the separation of interlocked objects requires a combination of the sliding and path planning methods discussed above, together with some enhancements – for example, a coarse boundary description of interlocked objects shows their edges as overlapping (as in the “two combs”

example in the last chapter). Reasoning about such interlocking objects would require a different approach to that used for coarse path planning, since there is no question of the whole object fitting between via points.

The extension of these techniques to real world fasteners would also require substantial additions to the qualitative representation used. Most real world fasteners involve three dimensional features, in addition to non-rigid behaviour. Another limitation is that motion in unfastening often includes rotation in addition to translation.

These problems have been noted by others (see especially Sedas and Talukdar [ST87]), and other treatments of fastening, whether using a qualitative approach or not, also tend to concentrate on two dimensional cases, without rotation, and assuming rigid objects. It is clear that the number of new issues involved in real world fastening mean that fastening analysis is considerably more complex than the type of tasks addressed by my implementation. It is interesting to note, however, that this qualitative technique has performance that is comparable to other systems addressing the same problem, while offering a number of advantages over those systems. These advantages are discussed further in the next chapter.

Chapter 6

Conclusions

The qualitative spatial reasoning methods described in this thesis can be evaluated from two different points of view. The first is the field of qualitative physics, where the significance of these methods lie in the fact that they can do things other qualitative reasoning systems cannot do. The second is the field of robotics, where the significance of the methods is that they provide new techniques for the type of reasoning that robots usually do.

From either point of view, it is necessary to define some criteria by which the usefulness of a new representation or reasoning method can be evaluated. This concluding chapter is divided into two main sections, each of which discusses the qualitative spatial reasoning methods in terms of a different set of evaluation criteria; the first section considers qualitative reasoning systems, and the second, robot reasoning systems.

6.1 Evaluating Spatial Qualitative Reasoning

Spatial reasoning, as a very basic human ability, has always been a priority in the investigation of AI reasoning systems. Hayes identified spatial reasoning as a particularly important component of the Naive Physics project, and spatial domains have often been chosen for experimental Qualitative Physics systems. This consensus on the importance of qualitative spatial reasoning has not, however, resulted in many qualitative systems that have general purpose spatial reasoning abilities.

The issue that most clearly separates the requirements of spatial reasoning from previous research in qualitative reasoning is the representation of *state*. Qualitative physics systems reason about their domains by identify-

ing and forecasting changes of state in a network of discrete devices. It is difficult to describe general motion in terms of change of state in a network, because free space is continuous, not discrete. Previous approaches to state-based qualitative spatial reasoning have involved the division of free space into a network of “discrete” regions, abstraction of physical devices so that they can be represented simply as linked nodes, or simplification of motion to mean only transitions between previously identified contact states.

Existing qualitative spatial reasoning systems have usually incorporated a preprocessing stage which uses conventional numeric techniques to determine possible qualitative motion states. The qualitative part of the system then analyses the dynamics of this qualitative description. The disadvantage of this technique is that the qualitative part of the system does not have direct access to a description of the scene geometry. The range of problems addressed by qualitative physics shows the consequences of using state information that does not include geometry – they can easily represent the *processes* involved in a mechanism (energy transfer, change of state), but they cannot reason about the simpler concepts of relative motion and constraint.

The partial distance ordering/extended polygon boundary representation of shape and position is a purely qualitative representation – it includes no quantitative information at all. It is at least as powerful for spatial description as the representations that have previously been used by qualitative reasoning systems in spatial domains. To illustrate this, the following section discusses three well-known domains for qualitative spatial reasoning, and shows how the PDO/EPB representation could be used in those domains.

6.1.1 Using PDO/EPB in Domains from Other Projects

In the roller coaster domain, motion of a block over the roller coaster can be described in the same terms as the sliding problem presented in the last chapter. The shape of the roller coaster, including discontinuities, can be represented as an extended polygon boundary. Relative slope of segments of the roller coaster can also be described. State can be represented, as in de Kleer’s system, as a combination of qualitative position on a particular segment of the roller coaster, and direction of motion. The only further information required to predict future motion is a representation of the external force acting on the system – gravity. In de Kleer’s system, gravity is implicit in the representation, but a reasoning system acting on the PDO/EPB rep-

resentation could explicitly include gravity as an influence that encourages closer proximity to the ground.

In the bouncing ball domain, it would also be necessary to include an explicit description of gravity – Forbus’ representation includes gravity as an implicit direction. The PDO/EPB representation can describe a moving object with real size and shape (whereas the bouncing ball in FROB is just a point mass), and it can describe flying, sliding, and collision, just as Forbus’ system does (motions like these are all discussed in the last chapter as ways of avoiding obstacles). The main advantage of this representation over the one used by Forbus is that it can describe change of qualitative state in the position of the ball without dividing space into problem-specific discrete regions. State change could be expressed in terms of changing relative proximity rather than absolute position, and this enables facilities such as describing the relative state of two bouncing balls, rather than just a single ball in a static world.

In the “mechanism world”, other qualitative analysis systems can describe only the motion of objects which are in contact. The PDO/EPB representation can be used to describe both motion in free space, and motion of objects in contact. The advantages of describing relative position globally are again apparent here, where the state of a mechanism may be a function of many individual parts in relative motion. This representation could therefore provide a basis for more general qualitative analysis of mechanisms, but it would have to be extended to include the influence of moving objects on other objects which they are in contact with. Such an extension could involve a qualitative version of the mechanics of pushing as analysed by Mason [Mas86]. The PDO/EPB representation does not provide the process description of energy transfer which is the central part of most mechanism analysis systems.

The facilities provided by the PDO/EPB representation can be used to carry out the spatial reasoning tasks associated with other qualitative reasoning systems that operate in these various spatial domains. The PDO/EPB methods do not require the use of a numeric preprocessing stage to carry out geometric analysis of possible motion and constraint, and the representation retains much of the spatial content of the scene – the scene geometry is not reduced to a discrete network.

6.1.2 General Evaluation of a Qualitative Representation

One of the main goals in developing the representation described in this thesis was to carry out spatial reasoning without any numerical geometric processing, and without reducing the spatial content of sensory information. The methods described in the last two chapters are successful to the degree that they can perform some reasoning of this type, but it is also necessary to evaluate whether they provide new capabilities when compared to existing systems, and how useful they can be in more general applications.

The evaluation of general purpose representations is not easy in any area of computer application, because it is difficult to separate considerations that apply uniquely to a given problem or domain from those that apply to any problem. In practice, such evaluation can only be carried out by testing the representation in a range of systems. The following list proposes a number of evaluation criteria which are important features of a good qualitative representation, but this list cannot be exhaustive until substantially more is known about qualitative reasoning.

Some evaluation criteria for a qualitative representation are:

1. What new facilities does the representation provide?
2. How wide is its application domain?
3. Does it seem to be intuitively accurate?
4. Can it be related to a known body of theory?
5. Is it consistent with other qualitative methods?
6. How does it perform in comparison to conventional methods?
7. Does it allow the integration of numeric information when necessary?
8. Is it easy to match and compare descriptions of states?
9. Is it easy to match and compare descriptions of processes?

When considered in terms of these criteria for evaluating qualitative representations, the following points can be noted about the PDO/EPB representation:

- PDO/EPB provides new qualitative methods for representing shape and relative position of objects. It can be used for spatial reasoning at a level that gives the reasoning system direct access to the scene description data in qualitative form. The description is at a level that is close to sensory data, and therefore includes no implicit functional information – from a qualitative point of view, it is purely a structural description.
- PDO/EPB is applicable in its present form to any two dimensional spatial reasoning problem. It can be used to plan and predict the effects of motion, but cannot directly represent information about process or influence.
- The extended polygon boundary description is a more intuitive way of describing an object boundary than methods which do not decompose shape into qualitative elements. As a description of objects in general, it does not really correspond to our intuitive impressions as well as the ASSF scheme – humans seem to deal with complete objects in terms of mass and area, rather than boundaries. The representation of relative position in terms of proximity is very intuitive.
- The type of geometry employed by a reasoning system that uses PDO/EPB is more similar to Euclidean geometry than Cartesian geometry. Most computational geometry systems employ Cartesian representations, so Cartesian geometry is more familiar in computer applications. There is, however, a large body of geometry theory that uses non-numeric Euclidean methods. Descriptions of shape in terms of changing proximity can make use of such methods.
- The PDO/EPB representation provides a two dimensional analogue to the quantity space by the use of the proximity ordering. For reasoning about motion and constraint, proximity is an important “quantity”. There are no “distinguished points” in the proximity quantity space, so it is constructed as a partial ordering for each scene, with no absolute values. The angle representation currently uses a simple linear quantity space. The types of operation carried out in both the direction and proximity spaces are normal qualitative reasoning techniques.
- The performance of the PDO/EPB representation in comparison to conventional numeric techniques is discussed in more detail in the next section, where conventional spatial reasoning methods for robots are

reconsidered. The qualitative methods are advantageous when data is inexact or missing, while a system with access to exact numeric data is able to solve many problems that a purely qualitative system could not.

- Numeric information can be integrated into the partial distance ordering, as discussed in chapter 4. Using a partial distance ordering with integrated numeric data would allow a reasoning system to use numeric techniques wherever appropriate. Flow of information between numeric and qualitative reasoning components could be achieved by creating or adjusting entries in the ordering.
- The qualitative state of a system under analysis is described solely by the proximity ordering. Changes in state resulting from perturbations to the system (motion of objects) are reflected in a new partial distance ordering. The system described in the previous chapter is able to carry out an envisionment process by postulating future states that can arise from the current one. State matching or comparison can be carried out simply by extracting a subset of the ordering that relates to objects of interest. The sliding system actually carried out state matching for recurring contact states across the whole global contact set.
- The reasoning system based on the PDO/EPB representation that is described in the last chapter maintains no explicit record of process. Process could be described as a sequence of states, but the process record would be completely implicit, making it very difficult to compare different processes. Explicit representation of process and influence, which are basic parts of qualitative physics, must be added as a different level of reasoning to the analysis of motion and constraint that can be carried out with PDO/EPB.

In summary, the PDO/EPB representation has most of the features that are expected of a representation for qualitative reasoning, and it can be applied to problems that have been used in the past to test qualitative spatial reasoning. It is able to answer a range of questions about object motion without making use of numeric data, by retaining geometric information in its qualitative scene description.

Although it is unable to explicitly represent influence or process (and cannot therefore be called a complete qualitative physics system), it does support a description of qualitative system state. The strategies used for

problem solving using the representation seem intuitive and natural to humans, which make them a good basis for “commonsense” reasoning systems.

6.2 Evaluating Qualitative Robot Reasoning

Very few attempts have been made (to date) to perform robot reasoning tasks using qualitative methods. Robot controllers receive information about the world in a numeric form from sensors, and they must supply precise numeric information in order to position and control actuators. Most established algorithms for analysing either sensory information or proposed motion are purely numeric, and most robots are programmed numerically (although the numeric nature of the program may be hidden from the programmer by the use of symbolic programming languages, or data capture devices such as “teach pendants”).

One of the few examples of an application in which a qualitative representation was used in a robot context is Burger and Bhanu’s system for qualitative motion understanding [BB87]. The qualitative representation used here was the output format of a sensory system which filtered information gathered by a vehicle moving over outdoor landscapes. The use of the term “qualitative” here implied mainly that the level of detail in the description is similar to that used by people – the system did not carry out qualitative reasoning of the type performed by qualitative physics systems.

There are no qualitative robot reasoning systems which can be used as a direct basis for comparison when evaluating the application of the PDO/EPB representation to robot reasoning, but the cases on which the PDO/EPB representation was tested (described in the last chapter) are very similar to some cases used to test experimental robot planning systems. One example is the disassembly planner described by Sedas and Talukdar [ST87]. This system plans how to remove components of an assembly from a constrained position into free space – essentially the same as the path planning problem formulation described in the last chapter. The PDO/EPB based system essentially solves the same problem, but does so without making use of numeric information.

There are three major advantages that the qualitative spatial reasoning methods presented in this thesis might provide when applied to robot problems. These are: the ability to operate with incomplete information, the ability to degrade gracefully, and the ability to support spatial reasoning at a level which is easily related to human problem solving. The following

three sections briefly discuss each of these.

6.2.1 Reasoning with Incomplete Information

Incomplete information should preferably be ignored during problem solving unless the missing information is necessary to solve the problem. The PDO/EPB representation provides for unknown information to be “hidden” (and thereby ignored) by the use of multiple levels of detail. Areas of a boundary which are irrelevant to a problem can be described extremely coarsely – perhaps just as a “wiggle”.

In the case where information is incomplete, but the missing quantities are required for the problem solution, the system must be able to hypothesize a constrained range of values for the unknown quantity. This ability was explained in the discussion of determination of fit in the path planning system. An unknown magnitude, whether a calculated “synthetic” magnitude or simply one that could not be measured, can be represented in the partial distance ordering as being completely unconstrained. The qualitative reasoning system can then make use of a repertoire of qualitative geometry techniques for constraining the value.

There are few conventional robot reasoning systems which are as readily able to operate with incomplete information. Firstly, numeric values are represented either as having a known value, or not having a value: there is no way of offering varying levels of detail to cover an intermittent lack of fine resolution. Secondly, if a hypothetical value is assigned to an unknown quantity, most systems cannot represent it as a range of plausible values, or even distinguish it from exact (measured) quantities. Some solid modelling systems do include facilities for representing tolerance information [RC86], and at least one robot motion planning system accounts for errors in part measurements or robot motion [Bro82a], but these numerical methods do not apply to as wide a range of problems as the PDO/EPB method.

The ability to operate with incomplete information can be useful in planning and control tasks if information cannot be obtained (for instance, the example given in the introduction: a partly hidden key which must be withdrawn by making use of a hypothetical description of the hidden portion). The ability to operate with hypothetical data is also an important facility in design tasks – the use of a partial distance ordering allows a designer to specify any (or no) constraints on an unknown value, and then continue normal operations as if the value had been specified exactly. A qualitative geometric reasoning system would be able to notify the designer as soon as

the constraints became insufficient or over-restrictive.

6.2.2 Providing Graceful Degradation

In the event that there is insufficient information for a robot reasoning system to continue with its task, it should be able to “degrade gracefully”. There are several components to graceful degradation (which is actually an objective for any well designed computer system):

- The system should still carry out any functions that can usefully be performed in the situation.
- It should inform its operators of the failure and suspected cause.
- It should remain capable of normal operation if improved information becomes available.
- Above all it should not “crash” or take incorrect actions.

There are graceful degradation facilities built into the PDO/EPB representation – in particular the partial distance ordering itself, and the “vague direction” handling features. In either directional or magnitude reasoning, information of varying precision can be accommodated, and the system is always aware of the possibility that the information available may not be sufficient. The system is able to distinguish between reasoning failures caused by insufficient information, and failures for other reasons, because all the qualitative geometry functions can return tokens to represent “unknown” as a valid result.

These facilities mean that the system can notify an operator of failure resulting from incomplete information. It can continue reasoning as normal, with the unknown token only propagating into areas that depend on the missing value. When this value does become known it cannot automatically be substituted back along the path where it was propagated (which might be an ideal for recovery after graceful degradation), but the parts of the reasoning process which need to be re-run can be directly identified.

Graceful degradation was one area in which the ASSF representation was noticeably inferior to the PDO/EPB representation. Because it depended on chains of relationships to establish relative positions or orientations of any two parts, it was unable to continue reasoning if any piece of information was missing. This handicap arose mainly because the way in which numeric descriptions were converted to qualitative ones was clumsy (especially

the use of axis length to create local quantity spaces, and the axis-relative location descriptions).

6.2.3 A Human Interface for Robot Programming

The ease with which a robot representation can be interpreted by humans is particularly important when the robots are dependent on humans to do some element of the reasoning involved in a task. This is not always considered in discussions of artificial intelligence for robots, especially where hypothetical autonomous robots (often anthropomorphic, and named “Robbie”) solve problems that are normally only encountered by humans.

The majority of today’s robots do not however “reason” about their workspace in any sense that we would recognise – they simply follow a prescribed sequence of motions (in some cases they may also react to unexpected events). A more realistic goal than the autonomous “Robbie” is task-level programming (which is actually a proposed level of cooperation in reasoning between a human and a robot) but even task-level programming requires a wide range of reasoning ability. The reasoning requirements of task-level programming include acquiring a description of the task from the programmer, interpreting information about the workspace from sensory information, planning actions, and using knowledge gained from previous tasks.

Spatial representations for robots developed to date have generally been specifically aimed at one of these functions. Programming representations have been developed from computer aided design methods or from general purpose computer languages, representations of sensory data have been designed specifically for particular sensors or object matching tasks, and motion planning systems have used ad hoc representations in order to apply particular algorithms for geometric collision avoidance or pathfinding.

The PDO/EPB representation may be well suited for use in task-level programming systems, for three reasons. Firstly, it describes things in a way that seem natural to a programmer – the “move towards” or “move forward and to the left” operations are the way that we naturally describe qualitative motion. Likewise, the description of shape seems natural to people – consider the following extended polygon boundary description of a light bulb: “Most of this shape is a circular curve, turning through about three-quarters of a circle. Each end of the curve extends into a wiggly section; the wiggles are parallel to each other. The last side is a complicated convex shape. It curves inward on each side, and curves outward in the middle”. This description

is easy for a human to construct, and is sufficient for qualitative spatial reasoning.

Secondly, the PDO/EPB representation can be used to reason about motion down to the level of individual robot movements. This is the range over which a unified representation is needed by systems such as Lozano-Perez's LAMA system. The robot movements are controlled according to "motion strategies" (INSERT, for example, which defines a strategy for inserting a cylinder into a round hole). The high level reasoning system need not have any information regarding the low level strategies, other than what effect they have. This is analagous to human motion – we plan our actions qualitatively, but individual motions use local feedback information, independent of that high level reasoning.

Thirdly, it would be possible to construct a PDO/EPB description directly from sensory data, so that a robot could update its internal representation of the world during performance of a task. The EPB shape representation is very similar to representations output by vision systems such as Mackerras' [Mac87b], and the proximity ordering transform can be carried out directly on a stored image after edge filtering and polygon segment identification.

The level of complexity in the qualitative PDO/EPB representation is therefore appropriate to the kind of reasoning problems that arise when a human must instruct a robot at the task level – that is, in human-like terms. The intuitive nature of the PDO/EPB representation is therefore as significant for robot applications as the advantages of graceful degradation, and of operation with incomplete data during reasoning.

6.2.4 Robot Reasoning with PDO/EPB

The PDO/EPB representation is capable of supporting at least some basic functions of a robot reasoning system; the path planning and sliding examples are typical (although simplified) problems in robotics. In solving these problems, the system based on the PDO/EPB representation demonstrates the ability to perform primitive qualitative reasoning about the effects of moving objects in the physical world.

This representation is sufficient to solve the kind of problems that humans expect to be able to solve using qualitative techniques (without measuring workpieces, for instance). Qualitative shape description is sufficient for spatial reasoning to be carried out at the commonsense level at which humans often operate, and it may therefore be sufficient for robot operation

to a human-like standard of performance.

The PDO/EPB representation was designed with robot applications in mind, and it incorporates the facilities recommended for robot reasoning in chapter 4. These facilities include reasoning in local contexts (decoupling object relationships in the scene using the proximity transform), multiple levels of detail (in the extended polygon boundary), and relative size description (in the partial distance ordering).

There are of course many limitations of qualitative techniques even when compared to current robot control methods. The loss of accurate positional information would not be acceptable for a robot that has any need to use geometric motion analysis techniques for instance. The methods presented also apply only to two dimensional cases; many three dimensional situations can be reduced to two dimensional motion, but most robots, even at the current level of robot technology, need to plan and operate in three dimensions.

6.3 Summary

The main achievement of the research described in this thesis has been the development of a representation to support qualitative spatial reasoning. The three key elements of this representation are the qualitative extended polygon boundary for shape description, the proximity transform for describing relative position, and the partial distance ordering for relating boundary size and proximity information.

The extended polygon boundary is a simple and effective qualitative description of shape, at a level which is easily constructed or interpreted by humans. An EPB shape description can include multiple levels of detail for individual portions of an object boundary, or for complete objects. The basic elements of the EPB description could be provided directly from a sensory system.

The proximity transform can extract useful qualitative information directly from two dimensional data describing a scene at a sensory (pixel or edge detection) level. It provides a basis for a quantity space which can describe two dimensional relative position and orientation. The distinction between “internal” and “external” proximity provides a consistent way of comparing shape to position.

The partial ordering is a well established technique for use in qualitative reasoning systems (although the quantity space is more often used in qualita-

tive physics). The application of a partial ordering to distance information, allowing the comparison of proximity information to object size, results in the partial distance ordering. This ordering has many important benefits for robust reasoning. The consistent treatment of feature size, distance between objects, and “synthetic” quantities resulting from geometric constructions, simplifies the implementation of qualitative geometric reasoning functions.

A qualitative proximity description of a scene constitutes a qualitative description of a unique state – a combination of objects in known relative positions. This state description can be used for envisionment purposes, without requiring any division of space into discrete regions. Envisionment of future states resulting from motion can be used for spatial reasoning problems such as path planning.

As a planning system, qualitative methods based on the PDO/EPB representation can carry out some tasks that are required in robot reasoning. The robust nature of the qualitative geometric reasoning provides the advantages for robotics of an ability to reason with incomplete information, and graceful degradation of the overall reasoning system. The human-like level at which this spatial reasoning is carried out gives it natural advantages for applications where humans interact with robot reasoning systems (such as task-level robot programming).

The PDO/EPB representation is a non-numeric technique for describing and reasoning about shape, space, position and motion at a simple geometric level. It has been developed together with qualitative spatial reasoning programs that can solve simple, but useful problems in spatial reasoning. It is an alternative approach to major reasoning tasks in qualitative mechanical physics and in robotics.

6.4 Future Research Directions

A number of desirable extensions to the current implementation of the PDO/EPB representation have been identified in the last two chapters. These include:

- Inclusion of order of magnitude information, and even optional numeric information, in the partial distance ordering.
- Partial ordering of angles, rather than using a simple quantity space.
- Integration of the system for reasoning about sliding (currently using the ASSF representation) with the free space motion reasoning system.

- Extension to three dimensions.

The presently described reasoning system can also be extended to become more useful either as a basis for qualitative physics, or for robotic reasoning:

- As a qualitative physics system, it would require a more explicit representation of process.
- A more complete investigation of qualitative reasoning as a basis for task-level programming would be particularly interesting, but would eventually require an interface to a real robot.
- The fastener analysis problems mentioned in the introduction to the thesis as a challenge to current technology in robotics may well benefit from qualitative reasoning, but fasteners, being real mechanical objects, could be more difficult to describe qualitatively than the “blocks world” objects of my examples.

Qualitative spatial reasoning is a feasible approach to a number of problems in artificial intelligence. With further refinement, the methods described in this thesis should prove useful to both robotics and qualitative physics.

Appendix A

EPB/PDO Representation Example

This appendix lists a complete description of a simple scene using the EPB/PDO representation, implemented in LISP. The scene described has been used as input for the path planning system. A scale drawing of the scene, with labels showing how boundary elements have been named, is in figure A.1.

The remainder of the appendix is a listing of the LISP representation.

Figure A.1: Scene for EPB/PDO Example (drawn to scale)

```

; The following code describes a scene with four objects in it,
; objects A,B,C and D.

; All atom names can be arbitrary, but for this example, they are
; named in the following mnemonic fashion:

; A segment atom is named with the object name, followed by a
; number; e.g. A3.

; A junction atom is named with the object name, followed by the
; numbers of the segments on either side; e.g. A3-4

; Object A has 14 sides, but can be regarded at coarser levels of
; detail, as represented by sides 15 to 20.
(putprop 'A '(A1 A2 A3 A4 A5 A6 A7 A8 A9 A10
              A11 A12 A13 A14 A15 A16 A17 A18 A19 A20
              A1-2 A2-3 A3-4 A4-5 A5-6 A6-7 A7-8 A8-9 A9-10
              A10-11 A11-12 A12-13 A13-14 A14-1 A15-16 A17-18 A18-19)
        'boundary)

; Objects B and C have four sides, and D has six:
(putprop 'B '(B1 B2 B3 B4 B1-2 B2-3 B3-4 B4-1) 'boundary)
(putprop 'C '(C1 C2 C3 C4 C1-2 C2-3 C3-4 C4-1) 'boundary)
(putprop 'D '(D1 D2 D3 D4 D5 D6 D1-2 D2-3 D3-4 D4-5
              D5-6 D6-1)
        'boundary)

; Note that default shape for a segment is a straight line
; (rather than curved or wiggly). No shape property therefore
; needs to be defined for a straight line, because assoc
; will return (curved nil), (wiggly nil) if the assoc list
; is empty.

;
; Segments on boundary of object A at the finest detail level:
;
(putprop 'A1 'A 'object)
(putprop 'A1 'segment 'type)
(putprop 'A1 '(A14-1) 'left)
(putprop 'A1 '(A1-2) 'right)
(putprop 'A1 'EQ9 'equidistant)
(putprop 'A1 '(I1 I3 I4 I41 I46 I53) 'internal)
(putprop 'A1 '(E1) 'external)

(putprop 'A2 'A 'object)
(putprop 'A2 'segment 'type)
(putprop 'A2 '(A1-2) 'left)
(putprop 'A2 '(A2-3) 'right)
(putprop 'A2 'EQ3a 'equidistant)
(putprop 'A2 '(I8) 'internal)
(putprop 'A2 '(E2) 'external)

(putprop 'A3 'A 'object)
(putprop 'A3 'segment 'type)
(putprop 'A3 '(A2-3) 'left)

```

```

(putprop 'A3 '(A3-4) 'right)
(putprop 'A3 'EQ1 'equidistant)
(putprop 'A3 '(I9 I9a I9b I9c I10 I11 I11a I44 I50) 'internal)
(putprop 'A3 '(E3) 'external)

(putprop 'A4 'A 'object)
(putprop 'A4 'segment 'type)
(putprop 'A4 '(A3-4) 'left)
(putprop 'A4 '(A4-5) 'right)
(putprop 'A4 'EQ5 'equidistant)
(putprop 'A4 '(I13 I14 I39 I51) 'internal)
(putprop 'A4 '(E4) 'external)

(putprop 'A5 'A 'object)
(putprop 'A5 'segment 'type)
(putprop 'A5 '(A4-5) 'left)
(putprop 'A5 '(A5-6) 'right)
(putprop 'A5 'EQ2 'equidistant)
; A5 has no internal proximities - it is down an alleyway
(putprop 'A5 '(E5) 'external)

(putprop 'A6 'A 'object)
(putprop 'A6 'segment 'type)
(putprop 'A6 '(A5-6) 'left)
(putprop 'A6 '(A6-7) 'right)
(putprop 'A6 'EQ3 'equidistant)
(putprop 'A6 '(I14 I38) 'internal)
(putprop 'A6 '(E6 E7 E8 E29) 'external)

(putprop 'A7 'A 'object)
(putprop 'A7 'segment 'type)
(putprop 'A7 '(A6-7) 'left)
(putprop 'A7 '(A7-8) 'right)
(putprop 'A7 'EQ1 'equidistant)
(putprop 'A7 '(I9c I28) 'internal)
; A7 has no external proximities - it is down an alleyway

(putprop 'A8 'A 'object)
(putprop 'A8 'segment 'type)
(putprop 'A8 '(A7-8) 'left)
(putprop 'A8 '(A8-9) 'right)
(putprop 'A8 'EQ1a 'equidistant)
(putprop 'A8 '(I18) 'internal)
(putprop 'A8 '(E8) 'external)

(putprop 'A9 'A 'object)
(putprop 'A9 'segment 'type)
(putprop 'A9 '(A8-9) 'left)
(putprop 'A9 '(A9-10) 'right)
(putprop 'A9 'EQ1 'equidistant)
;internal and external alleyways

(putprop 'A10 'A 'object)
(putprop 'A10 'segment 'type)
(putprop 'A10 '(A9-10) 'left)

```

```

(putprop 'A10 '(A10-11) 'right)
(putprop 'A10 'EQ1a 'equidistant)
(putprop 'A10 '(I18) 'internal)
(putprop 'A10 '(E10) 'external)

(putprop 'A11 'A 'object)
(putprop 'A11 'segment 'type)
(putprop 'A11 '(A10-11) 'left)
(putprop 'A11 '(A11-12) 'right)
(putprop 'A11 'EQ1 'equidistant)
(putprop 'A11 '(I3 I19 I4 I25) 'internal)

(putprop 'A12 'A 'object)
(putprop 'A12 'segment 'type)
(putprop 'A12 '(A11-12) 'left)
(putprop 'A12 '(A12-13) 'right)
(putprop 'A12 'EQ3 'equidistant)
(putprop 'A12 '(I20) 'internal)
(putprop 'A12 '(E10 E11 E6 E27) 'external)

(putprop 'A13 'A 'object)
(putprop 'A13 'segment 'type)
(putprop 'A13 '(A12-13) 'left)
(putprop 'A13 '(A13-14) 'right)
(putprop 'A13 'EQ6 'equidistant)
(putprop 'A13 '(I1 I23) 'internal)
(putprop 'A13 '(E12) 'external)

(putprop 'A14 'A 'object)
(putprop 'A14 'segment 'type)
(putprop 'A14 '(A13-14) 'left)
(putprop 'A14 '(A14-1) 'right)
(putprop 'A14 'EQ10 'equidistant)
(putprop 'A14 '(I20 I21 I22 I8 I36 I43 I54) 'internal)
(putprop 'A14 '(E13 E14) 'external)

;
; Junctions on boundary of object A at the fine detail level
;

(putprop 'A1-2 'A 'object)
(putprop 'A1-2 'junction 'type)
(putprop 'A1-2 '((angle right)
                (flex convex))
          'shape)
(putprop 'A1-2 '(A1) 'left)
(putprop 'A1-2 '(A2) 'right)
(putprop 'A1-2 '(I29) 'internal)
(putprop 'A1-2 '(E15 E15a) 'external)

(putprop 'A2-3 'A 'object)
(putprop 'A2-3 'junction 'type)
(putprop 'A2-3 '((angle right)
                (flex concave))
          'shape)

```

```

(putprop 'A2-3 '(A2) 'left)
(putprop 'A2-3 '(A3) 'right)
(putprop 'A2-3 '(I22 I19 I45) 'internal)
(putprop 'A2-3 '(E16) 'external)

(putprop 'A3-4 'A 'object)
(putprop 'A3-4 'junction 'type)
(putprop 'A3-4 '((angle right)
                 (flex convex))
          'shape)
(putprop 'A3-4 '(A3) 'left)
(putprop 'A3-4 '(A4) 'right)
(putprop 'A3-4 '(I30) 'internal)
(putprop 'A3-4 '(E17 E17a) 'external)

(putprop 'A4-5 'A 'object)
(putprop 'A4-5 'junction 'type)
(putprop 'A4-5 '((angle right)
                 (flex convex))
          'shape)
(putprop 'A4-5 '(A4 A16) 'left)
(putprop 'A4-5 '(A5 A20) 'right)
(putprop 'A4-5 '(I30 I31 I49) 'internal)
(putprop 'A4-5 '(E18 E18a) 'external)

(putprop 'A5-6 'A 'object)
(putprop 'A5-6 'junction 'type)
(putprop 'A5-6 '((angle right)
                 (flex convex))
          'shape)
(putprop 'A5-6 '(A5) 'left)
(putprop 'A5-6 '(A6 A17) 'right)
(putprop 'A5-6 '(I31) 'internal)
(putprop 'A5-6 '(E19 E19a) 'external)

(putprop 'A6-7 'A 'object)
(putprop 'A6-7 'junction 'type)
(putprop 'A6-7 '((angle right)
                 (flex concave))
          'shape)
(putprop 'A6-7 '(A6) 'left)
(putprop 'A6-7 '(A7) 'right)
(putprop 'A6-7 '(I10 I13 I37) 'internal)
(putprop 'A6-7 '(E20) 'external)

(putprop 'A7-8 'A 'object)
(putprop 'A7-8 'junction 'type)
(putprop 'A7-8 '((angle right)
                 (flex concave))
          'shape)
(putprop 'A7-8 '(A7) 'left)
(putprop 'A7-8 '(A8) 'right)
(putprop 'A7-8 '(I9b I27 I32) 'internal)
(putprop 'A7-8 '(E20) 'external)

```



```

(putprop 'A8-9 'A 'object)
(putprop 'A8-9 'junction 'type)
(putprop 'A8-9 '((angle right)
                (flex convex))
         'shape)
(putprop 'A8-9 '(A8) 'left)
(putprop 'A8-9 '(A9) 'right)
(putprop 'A8-9 '(I33) 'internal)
(putprop 'A8-9 '(E7) 'external)

(putprop 'A9-10 'A 'object)
(putprop 'A9-10 'junction 'type)
(putprop 'A9-10 '((angle right)
                (flex convex))
         'shape)
(putprop 'A9-10 '(A9) 'left)
(putprop 'A9-10 '(A10) 'right)
(putprop 'A9-10 '(I33) 'internal)
(putprop 'A9-10 '(E11) 'external)

(putprop 'A10-11 'A 'object)
(putprop 'A10-11 'junction 'type)
(putprop 'A10-11 '((angle right)
                 (flex concave))
         'shape)
(putprop 'A10-11 '(A10) 'left)
(putprop 'A10-11 '(A11) 'right)
(putprop 'A10-11 '(I9a I26 I32) 'internal)
(putprop 'A10-11 '(E21) 'external)

(putprop 'A11-12 'A 'object)
(putprop 'A11-12 'junction 'type)
(putprop 'A11-12 '((angle right)
                 (flex concave))
         'shape)
(putprop 'A11-12 '(A11) 'left)
(putprop 'A11-12 '(A12) 'right)
(putprop 'A11-12 '(I3 I21 I24) 'internal)
(putprop 'A11-12 '(E21) 'external)

(putprop 'A12-13 'A 'object)
(putprop 'A12-13 'junction 'type)
(putprop 'A12-13 '((angle right)
                 (flex convex))
         'shape)
(putprop 'A12-13 '(A12 A19) 'left)
(putprop 'A12-13 '(A13) 'right)
(putprop 'A12-13 '(I34) 'internal)
(putprop 'A12-13 '(E22 E19a) 'external)

(putprop 'A13-14 'A 'object)
(putprop 'A13-14 'junction 'type)
(putprop 'A13-14 '((angle right)
                 (flex convex))
         'shape)

```

```

(putprop 'A13-14 '(A13 A20) 'left)
(putprop 'A13-14 '(A14) 'right)
(putprop 'A13-14 '(I34 I35) 'internal)
(putprop 'A13-14 '(E23 E23a) 'external)

(putprop 'A14-1 'A 'object)
(putprop 'A14-1 'junction 'type)
(putprop 'A14-1 '((angle right)
                  (flex convex))
          'shape)
(putprop 'A14-1 '(A14) 'left)
(putprop 'A14-1 '(A1 A15) 'right)
(putprop 'A14-1 '(I35 I29 I48) 'internal)
(putprop 'A14-1 '(E24) 'external)

; Segments on boundary of object A at coarse detail levels:
;
(putprop 'A15 'A 'object)
(putprop 'A15 'segment 'type)
(putprop 'A15 '(A14-1) 'left)
(putprop 'A15 '(A15-16) 'right)
(putprop 'A15 'EQ10 'equidistant)
(putprop 'A15 '(I23 I24 I25 I26 I27 I28 I42 I47 I55) 'internal)
(putprop 'A15 '(E25) 'external)

(putprop 'A16 'A 'object)
(putprop 'A16 'segment 'type)
(putprop 'A16 '(A15-16) 'left)
(putprop 'A16 '(A4-5) 'right)
(putprop 'A16 'EQ10 'equidistant)
(putprop 'A16 '(I36 I37 I38 I40 I52) 'internal)
(putprop 'A16 '(E26) 'external)

(putprop 'A17 'A 'object)
(putprop 'A17 'segment 'type)
(putprop 'A17 '(A5-6) 'left)
(putprop 'A17 '(A17-18) 'right)
(putprop 'A17 'EQ1a 'equidistant)
(putprop 'A17 '(I39 I40) 'internal)
(putprop 'A17 '(E27 E28) 'external)

(putprop 'A18 'A 'object)
(putprop 'A18 'segment 'type)
(putprop 'A18 '(A17-18) 'left)
(putprop 'A18 '(A18-19) 'right)
(putprop 'A18 'EQ2 'equidistant)
(putprop 'A18 '(I41 I42) 'internal)
; In an alleyway - no externals

(putprop 'A19 'A 'object)
(putprop 'A19 'segment 'type)
(putprop 'A19 '(A18-19) 'left)
(putprop 'A19 '(A12-13) 'right)

```

```

(putprop 'A19 'EQ1a 'equidistant)
(putprop 'A19 '(I43) 'internal)
(putprop 'A19 '(E29 E28) 'external)

(putprop 'A20 'A 'object)
(putprop 'A20 'segment 'type)
(putprop 'A20 '(A4-5) 'left)
(putprop 'A20 '(A13-14) 'right)
(putprop 'A20 'EQ10 'equidistant)
(putprop 'A20 '(I44 I45 I46 I47) 'internal)
(putprop 'A20 '(E30) 'external)

;
; Junctions that occur between coarse segments only:
;

(putprop 'A15-16 'A 'object)
(putprop 'A15-16 'junction 'type)
(putprop 'A15-16 '((angle right)
                  (flex convex))
          'shape)
(putprop 'A15-16 '(A15) 'left)
(putprop 'A15-16 '(A16) 'right)
(putprop 'A15-16 '(I48 I49) 'internal)
(putprop 'A15-16 '(E31 E31a) 'external)

(putprop 'A17-18 'A 'object)
(putprop 'A17-18 'junction 'type)
(putprop 'A17-18 '((angle right)
                  (flex concave))
          'shape)
(putprop 'A17-18 '(A17) 'left)
(putprop 'A17-18 '(A18) 'right)
(putprop 'A17-18 '(I50 I51 I52) 'internal)
(putprop 'A17-18 '(E32) 'external)

(putprop 'A18-19 'A 'object)
(putprop 'A18-19 'junction 'type)
(putprop 'A18-19 '((angle right)
                  (flex concave))
          'shape)
(putprop 'A18-19 '(A18) 'left)
(putprop 'A18-19 '(A19) 'right)
(putprop 'A18-19 '(I50 I51 I52) 'internal)
(putprop 'A18-19 '(E32) 'external)

```

```

; The following is a description of shape B:
;
(putprop 'B1 'B 'object)
(putprop 'B1 'segment 'type)
(putprop 'B1 '(B4-1) 'left)
(putprop 'B1 '(B1-2) 'right)
(putprop 'B1 'EQ20 'equidistant)
(putprop 'B1 '(I56) 'internal)

(putprop 'B2 'B 'object)
(putprop 'B2 'segment 'type)
(putprop 'B2 '(B1-2) 'left)
(putprop 'B2 '(B2-3) 'right)
(putprop 'B2 'EQ4 'equidistant)
(putprop 'B2 '(I57) 'internal)

(putprop 'B3 'B 'object)
(putprop 'B3 'segment 'type)
(putprop 'B3 '(B2-3) 'left)
(putprop 'B3 '(B3-4) 'right)
(putprop 'B3 'EQ20 'equidistant)
(putprop 'B3 '(I56) 'internal)
(putprop 'B3 '(E33 E34 E35 E17 E31 E25 E16 E15 E1 E24 E37 E38 E39)
'external)

(putprop 'B4 'B 'object)
(putprop 'B4 'segment 'type)
(putprop 'B4 '(B3-4) 'left)
(putprop 'B4 '(B4-1) 'right)
(putprop 'B4 'EQ4 'equidistant)
(putprop 'B4 '(I57) 'internal)

(putprop 'B1-2 'B 'object)
(putprop 'B1-2 'junction 'type)
(putprop 'B1-2 '((angle right)
(flex convex))
'shape)
(putprop 'B1-2 '(B1) 'left)
(putprop 'B1-2 '(B2) 'right)
(putprop 'B1-2 '(I58 I59) 'internal)

(putprop 'B2-3 'B 'object)
(putprop 'B2-3 'junction 'type)
(putprop 'B2-3 '((angle right)
(flex convex))
'shape)
(putprop 'B2-3 '(B2) 'left)
(putprop 'B2-3 '(B3) 'right)
(putprop 'B2-3 '(I59 I60) 'internal)
(putprop 'B2-3 '(E40) 'external)

(putprop 'B3-4 'B 'object)
(putprop 'B3-4 'junction 'type)

```

```

(putprop 'B3-4 '((angle right)
                (flex convex))
         'shape)
(putprop 'B3-4 '(B3) 'left)
(putprop 'B3-4 '(B4) 'right)
(putprop 'B3-4 '(I60 I61) 'internal)
(putprop 'B3-4 '(E41) 'external)

(putprop 'B4-1 'B 'object)
(putprop 'B4-1 'junction 'type)
(putprop 'B4-1 '((angle right)
                (flex convex))
         'shape)
(putprop 'B4-1 '(B4) 'left)
(putprop 'B4-1 '(B1) 'right)
(putprop 'B4-1 '(I58 I61) 'internal)

;
; The following is a description of shape C:
;
(putprop 'C1 'C 'object)
(putprop 'C1 'segment 'type)
(putprop 'C1 '(C4-1) 'left)
(putprop 'C1 '(C1-2) 'right)
(putprop 'C1 'EQ10 'equidistant)
(putprop 'C1 '(I62) 'internal)
(putprop 'C1 '(E33) 'external)

(putprop 'C2 'C 'object)
(putprop 'C2 'segment 'type)
(putprop 'C2 '(C1-2) 'left)
(putprop 'C2 '(C2-3) 'right)
(putprop 'C2 'EQ17 'equidistant)
(putprop 'C2 '(I63) 'internal)

(putprop 'C3 'C 'object)
(putprop 'C3 'segment 'type)
(putprop 'C3 '(C2-3) 'left)
(putprop 'C3 '(C3-4) 'right)
(putprop 'C3 'EQ10 'equidistant)
(putprop 'C3 '(I62) 'internal)

(putprop 'C4 'C 'object)
(putprop 'C4 'segment 'type)
(putprop 'C4 '(C3-4) 'left)
(putprop 'C4 '(C4-1) 'right)
(putprop 'C4 'EQ17 'equidistant)
(putprop 'C4 '(I63) 'internal)
(putprop 'C4 '(E42 E43 E44 E18a E26 E4 E17a E31a E15a) 'external)

(putprop 'C1-2 'C 'object)
(putprop 'C1-2 'junction 'type)
(putprop 'C1-2 '((angle right)
                (flex convex))
         'shape)

```

```

(putprop 'C1-2 '(C1) 'left)
(putprop 'C1-2 '(C2) 'right)
(putprop 'C1-2 '(I64 I65) 'internal)
(putprop 'C1-2 '(E40) 'external)

(putprop 'C2-3 'C 'object)
(putprop 'C2-3 'junction 'type)
(putprop 'C2-3 '((angle right)
                 (flex convex))
          'shape)
(putprop 'C2-3 '(C2) 'left)
(putprop 'C2-3 '(C3) 'right)
(putprop 'C2-3 '(I65 I66) 'internal)

(putprop 'C3-4 'C 'object)
(putprop 'C3-4 'junction 'type)
(putprop 'C3-4 '((angle right)
                 (flex convex))
          'shape)
(putprop 'C3-4 '(C3) 'left)
(putprop 'C3-4 '(C4) 'right)
(putprop 'C3-4 '(I66 I67) 'internal)
(putprop 'C3-4 '(E45) 'external)

(putprop 'C4-1 'C 'object)
(putprop 'C4-1 'junction 'type)
(putprop 'C4-1 '((angle right)
                 (flex convex))
          'shape)
(putprop 'C4-1 '(C4) 'left)
(putprop 'C4-1 '(C1) 'right)
(putprop 'C4-1 '(I64 I67) 'internal)
(putprop 'C4-1 '(E34) 'external)

;
; The following is a description of shape D:
;
(putprop 'D1 'D 'object)
(putprop 'D1 'segment 'type)
(putprop 'D1 '(D6-1) 'left)
(putprop 'D1 '(D1-2) 'right)
(putprop 'D1 'EQ4 'equidistant)
(putprop 'D1 '(I68) 'internal)
(putprop 'D1 '(E39) 'external)

(putprop 'D2 'D 'object)
(putprop 'D2 'segment 'type)
(putprop 'D2 '(D1-2) 'left)
(putprop 'D2 '(D2-3) 'right)
(putprop 'D2 'EQ14 'equidistant)
(putprop 'D2 '(I69) 'internal)
(putprop 'D2 '(E13 E23 E44) 'external)

(putprop 'D3 'D 'object)
(putprop 'D3 'segment 'type)

```

```

(putprop 'D3 '(D2-3) 'left)
(putprop 'D3 '(D3-4) 'right)
(putprop 'D3 'EQ15a 'equidistant)
(putprop 'D3 '(I70) 'internal)
(putprop 'D3 '(E37 E12 E22 E19 E5) 'external)

(putprop 'D4 'D 'object)
(putprop 'D4 'segment 'type)
(putprop 'D4 '(D3-4) 'left)
(putprop 'D4 '(D4-5) 'right)
(putprop 'D4 'EQ4 'equidistant)
(putprop 'D4 '(I71) 'internal)
(putprop 'D4 '(E42) 'external)

(putprop 'D5 'D 'object)
(putprop 'D5 'segment 'type)
(putprop 'D5 '(D4-5) 'left)
(putprop 'D5 '(D5-6) 'right)
(putprop 'D5 'EQ16a 'equidistant)
(putprop 'D5 '(I68 I70) 'internal)

(putprop 'D6 'D 'object)
(putprop 'D6 'segment 'type)
(putprop 'D6 '(D5-6) 'left)
(putprop 'D6 '(D6-1) 'right)
(putprop 'D6 'EQ15 'equidistant)
(putprop 'D6 '(I69 I71) 'internal)

(putprop 'D1-2 'D 'object)
(putprop 'D1-2 'junction 'type)
(putprop 'D1-2 '((angle right)
                 (flex convex))
          'shape)
(putprop 'D1-2 '(D1) 'left)
(putprop 'D1-2 '(D2) 'right)
(putprop 'D1-2 '(I72) 'internal)
(putprop 'D1-2 '(E38 E14) 'external)

(putprop 'D2-3 'D 'object)
(putprop 'D2-3 'junction 'type)
(putprop 'D2-3 '((angle right)
                 (flex concave))
          'shape)
(putprop 'D2-3 '(D2) 'left)
(putprop 'D2-3 '(D3) 'right)
(putprop 'D2-3 '(I73) 'internal)
(putprop 'D2-3 '(E23a) 'external)

(putprop 'D3-4 'D 'object)
(putprop 'D3-4 'junction 'type)
(putprop 'D3-4 '((angle right)
                 (flex convex))
          'shape)
(putprop 'D3-4 '(D3) 'left)
(putprop 'D3-4 '(D4) 'right)

```

```

(putprop 'D3-4 '(I74) 'internal)
(putprop 'D3-4 '(E43) 'external)

(putprop 'D4-5 'D 'object)
(putprop 'D4-5 'junction 'type)
(putprop 'D4-5 '((angle right)
                (flex convex))
          'shape)
(putprop 'D4-5 '(D4) 'left)
(putprop 'D4-5 '(D5) 'right)
(putprop 'D4-5 '(I74 I75) 'internal)
(putprop 'D4-5 '(E45) 'external)

(putprop 'D5-6 'D 'object)
(putprop 'D5-6 'junction 'type)
(putprop 'D5-6 '((angle right)
                (flex convex))
          'shape)
(putprop 'D5-6 '(D5) 'left)
(putprop 'D5-6 '(D6) 'right)
(putprop 'D5-6 '(I73 I75 I76) 'internal)

(putprop 'D6-1 'D 'object)
(putprop 'D6-1 'junction 'type)
(putprop 'D6-1 '((angle right)
                (flex convex))
          'shape)
(putprop 'D6-1 '(D6) 'left)
(putprop 'D6-1 '(D1) 'right)
(putprop 'D6-1 '(I76 I72) 'internal)
(putprop 'D6-1 '(E41) 'external)

```



```

;
; Internal proximity pairs for the whole scene:
;
(setf I1 '(A1 A13))
(setf I3 '(A1 A11-12))
(setf I4 '(A1 A11))

(setf I8 '(A2 A14))

(setf I9 '(A3 A11))
(setf I9a '(A3 A10-11))
(setf I9b '(A3 A7-8))
(setf I9c '(A3 A7))
(setf I10 '(A3 A6-7))
(setf I11 '(A3 A18))
(setf I11a '(A3 A17-18))

(setf I13 '(A4 A6-7))
(setf I14 '(A4 A6))

(setf I18 '(A8 A10))

(setf I19 '(A11 A2-3))

(setf I20 '(A12 A14))

(setf I21 '(A14 A11-12))
(setf I22 '(A14 A2-3))

(setf I23 '(A15 A13))
(setf I24 '(A15 A11-12))
(setf I25 '(A15 A11))
(setf I26 '(A15 A10-11))
(setf I27 '(A15 A7-8))
(setf I28 '(A15 A7))

(setf I29 '(A1-2 A14-1))

(setf I30 '(A3-4 A4-5))

(setf I31 '(A4-5 A5-6))

(setf I32 '(A7-8 A10-11))

(setf I33 '(A8-9 A9-10))

(setf I34 '(A12-13 A13-14))

(setf I35 '(A13-14 A14-1))

(setf I36 '(A16 A14))
(setf I37 '(A16 A6-7))
(setf I38 '(A16 A6))

(setf I39 '(A17 A4))
(setf I40 '(A17 A16))

(setf I41 '(A18 A1))
(setf I42 '(A18 A15))

(setf I43 '(A19 A14))

(setf I44 '(A20 A3))
(setf I45 '(A20 A2-3))

```

```

(setf I46 '(A20 A1))
(setf I47 '(A20 A15))

(setf I48 '(A15-16 A14-1))
(setf I49 '(A15-16 A4-5))

(setf I50 '(A17-18 A3))
(setf I51 '(A17-18 A4))
(setf I52 '(A17-18 A16))

(setf I53 '(A18-19 A1))
(setf I54 '(A18-19 A14))
(setf I55 '(A18-19 A15))

(setf I56 '(B1 B3))
(setf I57 '(B2 B4))

(setf I58 '(B1-2 B4-1))
(setf I59 '(B1-2 B2-3))
(setf I60 '(B2-3 B3-4))
(setf I61 '(B3-4 B4-1))

(setf I62 '(C1 C3))
(setf I63 '(C2 C4))

(setf I64 '(C1-2 C4-1))
(setf I65 '(C1-2 C2-3))
(setf I66 '(C2-3 C3-4))
(setf I67 '(C3-4 C4-1))

(setf I68 '(D1 D5))
(setf I69 '(D2 D6))
(setf I70 '(D3 D5))
(setf I71 '(D4 D6))

(setf I72 '(D1-2 D6-1))
(setf I73 '(D2-3 D5-6))
(setf I74 '(D3-4 D4-5))
(setf I75 '(D4-5 D5-6))
(setf I76 '(D5-6 D6-1))

; External proximity pairs:
;
(setf E1 '(A1 B3))
(setf E2 '(A2 C4))
(setf E3 '(A3 B3))
(setf E4 '(A4 C4))
(setf E5 '(A5 D3))
(setf E6 '(A6 A12))
(setf E7 '(A6 A8-9))
(setf E8 '(A6 A8))
(setf E10 '(A10 A12))
(setf E11 '(A12 A9-10))
(setf E12 '(A13 D3))
(setf E13 '(A14 D2))
(setf E14 '(A14 D1-2))
(setf E15 '(A1-2 B3))
(setf E15a '(A1-2 C4))
(setf E16 '(A2-3 B3))
(setf E17 '(A3-4 B3))

```

```
(setf E17a '(A3-4 C4))
(setf E18 '(A4-5 D3))
(setf E18a '(A4-5 C4))
(setf E19 '(A5-6 D3))
(setf E19a '(A5-6 A12-13))
(setf E20 '(A6-7 A7-8))
(setf E21 '(A10-11 A11-12))
(setf E22 '(A12-13 D3))
(setf E23 '(A13-14 D2))
(setf E23a '(A13-14 D2-3))
(setf E24 '(A14-1 B3))
(setf E25 '(A15 B3))
(setf E26 '(A16 C4))
(setf E27 '(A17 A12))
(setf E28 '(A17 A19))
(setf E29 '(A19 A6))
(setf E30 '(A20 D3))
(setf E31 '(A15-16 B3))
(setf E31a '(A15-16 C4))
(setf E32 '(A17-18 A18-19))

(setf E33 '(B3 C1))
(setf E34 '(B3 C4-1))
(setf E35 '(B3 D3-4))
(setf E37 '(B3 D3))
(setf E38 '(B3 D1-2))
(setf E39 '(B3 D1))
(setf E40 '(B2-3 C1-2))
(setf E41 '(B3-4 D6-1))

(setf E42 '(C4 D4))
(setf E43 '(C4 D3-4))
(setf E44 '(C4 D2))
(setf E45 '(C3-4 D4-5))
```

```

;
; Equidistance lists define which proximities and segment
; sizes are equal:
;
(setf contact '(E33 E34 E40))

(setf EQ1 '(E8 E21 E10 E20 E7 E11 I33 I32 I18 A3 A7 A9 A11))
(setf EQ1a '(A8 A10 A17 A19))
(setf EQ2 '(E32 E29 E27 E28 E19a E6 I14 I13 I31 I37 I38 I51
            I52 I40 I39 A5 A18))
(setf EQ3 '(E42 E43 E45 A6 A12))
(setf EQ3a '(I9 I9a I9b I9c I10 I19 I50 A2))
(setf EQ4 '(I11 I11a I56 I59 I61 I69 I70 I72 I73 I74 E23 E13
            E14 B2 B4 D1 D4))
(setf EQ5 '(I30 I45 I44 A4))
(setf EQ6 '(I34 I20 I21 I43 I54 A13))
(setf EQ7 '(I3 I4 I24 I25 I26 I27 I28))
(setf EQ8 '(I53 I55 I41 I42))
(setf EQ9 '(I22 I8 I29 A1))
(setf EQ10 '(I63 I64 I66 I46 I47 I48 I49 I36 I35 I1 I23 E1
            E15 E24 E25 E31 A14 A15 A16 A20 C1 C3))
(setf EQ11 '(E3 E16 E17))
(setf EQ12 '(E38 E39 E41))
(setf EQ13 '(E30 E22 E19 E18 E12 E5 E23a))
(setf EQ14 '(E31a E26 E18a E17a E4 D2))
(setf EQ14a '(E2 E15a))
(setf EQ15 '(I76 I68 D6))
(setf EQ15a '(D3))
(setf EQ16 '(E37 E35 E44))
(setf EQ16a '(I71 I75 D5))
(setf EQ17 '(I67 I65 I62 C2 C4))
(setf EQ20 '(I60 I58 I57 B1 B3))

```

```
; Pointers from external proximity pairs to associated  
; equidistance lists:  
;
```

```
(putprop 'E33 'contact 'equidistant)  
(putprop 'E34 'contact 'equidistant)  
(putprop 'E40 'contact 'equidistant)  
(putprop 'E8 'EQ1 'equidistant)  
(putprop 'E21 'EQ1 'equidistant)  
(putprop 'E10 'EQ1 'equidistant)  
(putprop 'E20 'EQ1 'equidistant)  
(putprop 'E7 'EQ1 'equidistant)  
(putprop 'E11 'EQ1 'equidistant)  
(putprop 'E32 'EQ2 'equidistant)  
(putprop 'E29 'EQ2 'equidistant)  
(putprop 'E27 'EQ2 'equidistant)  
(putprop 'E28 'EQ2 'equidistant)  
(putprop 'E19a 'EQ2 'equidistant)  
(putprop 'E6 'EQ2 'equidistant)  
(putprop 'E42 'EQ3 'equidistant)  
(putprop 'E43 'EQ3 'equidistant)  
(putprop 'E45 'EQ3 'equidistant)  
(putprop 'E23 'EQ4 'equidistant)  
(putprop 'E13 'EQ4 'equidistant)  
(putprop 'E14 'EQ4 'equidistant)  
(putprop 'E1 'EQ10 'equidistant)  
(putprop 'E15 'EQ10 'equidistant)  
(putprop 'E24 'EQ10 'equidistant)  
(putprop 'E25 'EQ10 'equidistant)  
(putprop 'E31 'EQ10 'equidistant)  
(putprop 'E3 'EQ11 'equidistant)  
(putprop 'E16 'EQ11 'equidistant)  
(putprop 'E17 'EQ11 'equidistant)  
(putprop 'E38 'EQ12 'equidistant)  
(putprop 'E39 'EQ12 'equidistant)  
(putprop 'E41 'EQ12 'equidistant)  
(putprop 'E30 'EQ13 'equidistant)  
(putprop 'E22 'EQ13 'equidistant)  
(putprop 'E19 'EQ13 'equidistant)  
(putprop 'E18 'EQ13 'equidistant)  
(putprop 'E12 'EQ13 'equidistant)  
(putprop 'E23a 'EQ13 'equidistant)  
(putprop 'E5 'EQ13 'equidistant)  
(putprop 'E31a 'EQ14 'equidistant)  
(putprop 'E26 'EQ14 'equidistant)  
(putprop 'E18a 'EQ14 'equidistant)  
(putprop 'E17a 'EQ14 'equidistant)  
(putprop 'E4 'EQ14 'equidistant)  
(putprop 'E2 'EQ14a 'equidistant)  
(putprop 'E15a 'EQ14a 'equidistant)  
(putprop 'E37 'EQ16 'equidistant)
```

```

(putprop 'E35 'EQ16 'equidistant)
(putprop 'E44 'EQ16 'equidistant)
;
; Pointers from internal proximity pairs to equidistance lists:
;
(putprop 'I33 'EQ1 'equidistant)
(putprop 'I32 'EQ1 'equidistant)
(putprop 'I18 'EQ1 'equidistant)
(putprop 'I14 'EQ2 'equidistant)
(putprop 'I13 'EQ2 'equidistant)
(putprop 'I31 'EQ2 'equidistant)
(putprop 'I37 'EQ2 'equidistant)
(putprop 'I38 'EQ2 'equidistant)
(putprop 'I51 'EQ2 'equidistant)
(putprop 'I52 'EQ2 'equidistant)
(putprop 'I40 'EQ2 'equidistant)
(putprop 'I39 'EQ2 'equidistant)
(putprop 'I9 'EQ3a 'equidistant)
(putprop 'I9a 'EQ3a 'equidistant)
(putprop 'I9b 'EQ3a 'equidistant)
(putprop 'I9c 'EQ3a 'equidistant)
(putprop 'I10 'EQ3a 'equidistant)
(putprop 'I19 'EQ3a 'equidistant)
(putprop 'I50 'EQ3a 'equidistant)
(putprop 'I11 'EQ4 'equidistant)
(putprop 'I11a 'EQ4 'equidistant)
(putprop 'I56 'EQ4 'equidistant)
(putprop 'I59 'EQ4 'equidistant)
(putprop 'I61 'EQ4 'equidistant)
(putprop 'I69 'EQ4 'equidistant)
(putprop 'I70 'EQ4 'equidistant)
(putprop 'I72 'EQ4 'equidistant)
(putprop 'I73 'EQ4 'equidistant)
(putprop 'I74 'EQ4 'equidistant)
(putprop 'I30 'EQ5 'equidistant)
(putprop 'I45 'EQ5 'equidistant)
(putprop 'I44 'EQ5 'equidistant)
(putprop 'I34 'EQ6 'equidistant)
(putprop 'I20 'EQ6 'equidistant)
(putprop 'I21 'EQ6 'equidistant)
(putprop 'I43 'EQ6 'equidistant)
(putprop 'I54 'EQ6 'equidistant)
(putprop 'I3 'EQ7 'equidistant)
(putprop 'I4 'EQ7 'equidistant)
(putprop 'I24 'EQ7 'equidistant)
(putprop 'I25 'EQ7 'equidistant)
(putprop 'I26 'EQ7 'equidistant)
(putprop 'I27 'EQ7 'equidistant)
(putprop 'I28 'EQ7 'equidistant)
(putprop 'I53 'EQ8 'equidistant)
(putprop 'I55 'EQ8 'equidistant)

```

(putprop 'I41 'EQ8 'equidistant)
(putprop 'I42 'EQ8 'equidistant)
(putprop 'I22 'EQ9 'equidistant)
(putprop 'I8 'EQ9 'equidistant)
(putprop 'I29 'EQ9 'equidistant)
(putprop 'I63 'EQ10 'equidistant)
(putprop 'I64 'EQ10 'equidistant)
(putprop 'I66 'EQ10 'equidistant)
(putprop 'I46 'EQ10 'equidistant)
(putprop 'I47 'EQ10 'equidistant)
(putprop 'I48 'EQ10 'equidistant)
(putprop 'I49 'EQ10 'equidistant)
(putprop 'I36 'EQ10 'equidistant)
(putprop 'I35 'EQ10 'equidistant)
(putprop 'I1 'EQ10 'equidistant)
(putprop 'I23 'EQ10 'equidistant)
(putprop 'I76 'EQ15 'equidistant)
(putprop 'I68 'EQ15 'equidistant)
(putprop 'I67 'EQ17 'equidistant)
(putprop 'I65 'EQ17 'equidistant)
(putprop 'I62 'EQ17 'equidistant)
(putprop 'I71 'EQ16a 'equidistant)
(putprop 'I75 'EQ16a 'equidistant)
(putprop 'I60 'EQ20 'equidistant)
(putprop 'I58 'EQ20 'equidistant)
(putprop 'I57 'EQ20 'equidistant)

```
; Note that the name Inn rather than Enn does not yet
; mean anything. We still have to describe whether each
; proximity is internal or external:
;
```

```
(putprop 'I1 'internal 'type)
(putprop 'I3 'internal 'type)
(putprop 'I4 'internal 'type)
(putprop 'I8 'internal 'type)
(putprop 'I9 'internal 'type)
(putprop 'I9a 'internal 'type)
(putprop 'I9b 'internal 'type)
(putprop 'I9c 'internal 'type)
(putprop 'I10 'internal 'type)
(putprop 'I11 'internal 'type)
(putprop 'I11a 'internal 'type)
(putprop 'I13 'internal 'type)
(putprop 'I14 'internal 'type)
(putprop 'I18 'internal 'type)
(putprop 'I19 'internal 'type)
(putprop 'I20 'internal 'type)
(putprop 'I21 'internal 'type)
(putprop 'I22 'internal 'type)
(putprop 'I23 'internal 'type)
(putprop 'I24 'internal 'type)
(putprop 'I25 'internal 'type)
(putprop 'I26 'internal 'type)
(putprop 'I27 'internal 'type)
(putprop 'I28 'internal 'type)
(putprop 'I29 'internal 'type)
(putprop 'I30 'internal 'type)
(putprop 'I31 'internal 'type)
(putprop 'I32 'internal 'type)
(putprop 'I33 'internal 'type)
(putprop 'I34 'internal 'type)
(putprop 'I35 'internal 'type)
(putprop 'I36 'internal 'type)
(putprop 'I37 'internal 'type)
(putprop 'I38 'internal 'type)
(putprop 'I39 'internal 'type)
(putprop 'I40 'internal 'type)
(putprop 'I41 'internal 'type)
(putprop 'I42 'internal 'type)
(putprop 'I43 'internal 'type)
(putprop 'I44 'internal 'type)
(putprop 'I45 'internal 'type)
(putprop 'I46 'internal 'type)
(putprop 'I47 'internal 'type)
(putprop 'I48 'internal 'type)
(putprop 'I49 'internal 'type)
(putprop 'I50 'internal 'type)
```


(putprop 'I51 'internal 'type)
(putprop 'I52 'internal 'type)
(putprop 'I53 'internal 'type)
(putprop 'I54 'internal 'type)
(putprop 'I55 'internal 'type)
(putprop 'I56 'internal 'type)
(putprop 'I57 'internal 'type)
(putprop 'I58 'internal 'type)
(putprop 'I59 'internal 'type)
(putprop 'I60 'internal 'type)
(putprop 'I61 'internal 'type)
(putprop 'I62 'internal 'type)
(putprop 'I63 'internal 'type)
(putprop 'I64 'internal 'type)
(putprop 'I65 'internal 'type)
(putprop 'I66 'internal 'type)
(putprop 'I67 'internal 'type)
(putprop 'I68 'internal 'type)
(putprop 'I69 'internal 'type)
(putprop 'I70 'internal 'type)
(putprop 'I71 'internal 'type)
(putprop 'I72 'internal 'type)
(putprop 'I73 'internal 'type)
(putprop 'I74 'internal 'type)
(putprop 'I75 'internal 'type)
(putprop 'I76 'internal 'type)

(putprop 'E1 'external 'type)
(putprop 'E2 'external 'type)
(putprop 'E3 'external 'type)
(putprop 'E4 'external 'type)
(putprop 'E5 'external 'type)
(putprop 'E6 'external 'type)
(putprop 'E7 'external 'type)
(putprop 'E8 'external 'type)
(putprop 'E10 'external 'type)
(putprop 'E11 'external 'type)
(putprop 'E12 'external 'type)
(putprop 'E13 'external 'type)
(putprop 'E14 'external 'type)
(putprop 'E15 'external 'type)
(putprop 'E15a 'external 'type)
(putprop 'E16 'external 'type)
(putprop 'E17 'external 'type)
(putprop 'E17a 'external 'type)
(putprop 'E18 'external 'type)
(putprop 'E18a 'external 'type)
(putprop 'E19 'external 'type)
(putprop 'E19a 'external 'type)
(putprop 'E20 'external 'type)
(putprop 'E21 'external 'type)
(putprop 'E22 'external 'type)

(putprop 'E23 'external 'type)
(putprop 'E23a 'external 'type)
(putprop 'E24 'external 'type)
(putprop 'E25 'external 'type)
(putprop 'E26 'external 'type)
(putprop 'E27 'external 'type)
(putprop 'E28 'external 'type)
(putprop 'E29 'external 'type)
(putprop 'E30 'external 'type)
(putprop 'E31 'external 'type)
(putprop 'E31a 'external 'type)
(putprop 'E32 'external 'type)
(putprop 'E33 'external 'type)
(putprop 'E34 'external 'type)
(putprop 'E35 'external 'type)
(putprop 'E37 'external 'type)
(putprop 'E38 'external 'type)
(putprop 'E39 'external 'type)
(putprop 'E40 'external 'type)
(putprop 'E41 'external 'type)
(putprop 'E42 'external 'type)
(putprop 'E43 'external 'type)
(putprop 'E44 'external 'type)
(putprop 'E45 'external 'type)

```
; Finally, even though the names for equal distances are
; already alphabetically ordered, the partial ordering must
; be established in lisp:
```

```
(putprop 'contact 'EQ1 'larger)
(putprop 'EQ1 'contact 'smaller)
(putprop 'EQ1 'EQ1a 'larger)
(putprop 'EQ1a 'EQ1 'smaller)
(putprop 'EQ1a 'EQ2 'larger)
(putprop 'EQ2 'EQ1a 'smaller)
(putprop 'EQ2 'EQ3 'larger)
(putprop 'EQ3 'EQ2 'smaller)
(putprop 'EQ3 'EQ3a 'larger)
(putprop 'EQ3a 'EQ3 'smaller)
(putprop 'EQ3a 'EQ4 'larger)
(putprop 'EQ4 'EQ3a 'smaller)
(putprop 'EQ4 'EQ5 'larger)
(putprop 'EQ5 'EQ4 'smaller)
(putprop 'EQ5 'EQ6 'larger)
(putprop 'EQ6 'EQ5 'smaller)
(putprop 'EQ6 'EQ7 'larger)
(putprop 'EQ7 'EQ6 'smaller)
(putprop 'EQ7 'EQ8 'larger)
(putprop 'EQ8 'EQ7 'smaller)
(putprop 'EQ8 'EQ9 'larger)
(putprop 'EQ9 'EQ8 'smaller)
(putprop 'EQ9 'EQ10 'larger)
(putprop 'EQ10 'EQ9 'smaller)
(putprop 'EQ10 'EQ11 'larger)
(putprop 'EQ11 'EQ10 'smaller)
(putprop 'EQ11 'EQ12 'larger)
(putprop 'EQ12 'EQ11 'smaller)
(putprop 'EQ12 'EQ13 'larger)
(putprop 'EQ13 'EQ12 'smaller)
(putprop 'EQ13 'EQ14 'larger)
(putprop 'EQ14 'EQ13 'smaller)
(putprop 'EQ14 'EQ14a 'larger)
(putprop 'EQ14a 'EQ14 'smaller)
(putprop 'EQ14a 'EQ15 'larger)
(putprop 'EQ15 'EQ14a 'smaller)
(putprop 'EQ15 'EQ15a 'larger)
(putprop 'EQ15a 'EQ15 'smaller)
(putprop 'EQ15a 'EQ16 'larger)
(putprop 'EQ16 'EQ15a 'smaller)
(putprop 'EQ16 'EQ16a 'larger)
(putprop 'EQ16a 'EQ16 'smaller)
(putprop 'EQ16a 'EQ17 'larger)
(putprop 'EQ17 'EQ16a 'smaller)
(putprop 'EQ17 'EQ20 'larger)
(putprop 'EQ20 'EQ17 'smaller)
```

```
(putprop 'EQ17 'EQ20 'larger)
(putprop 'EQ20 'EQ17 'smaller)
```

Appendix B

Robot Fastening and Disassembly Projects

The early stages of this research involved an investigation into the feasibility of systems to solve two problems in robotics. The first was a general approach to acquiring assembly descriptions by disassembling. The second was a general method for reasoning about mechanical fastening (necessary to solve the disassembly problem in real-world situations). The investigation of qualitative reasoning described in the body of the thesis arose from the latter of these two topics, but there are many other interesting projects that might be carried out in studying this overall area.

This appendix contains a “navigation chart” which I constructed at the start of my research to show the relationships between these possible projects, ranging from abstract geometric reasoning to physical control of robot manipulators. The chart has been included here both to illustrate my view of the wider context of this project, and also as a resource for readers who have a further interest in the disassembly/fastening problem.

Bibliography

- [AB85] P. Allen and R. Bajcsy. Object recognition using vision and touch. In *Proceedings International Joint Conference on Artificial Intelligence*, 1985.
- [ABB⁺75] A. P. Ambler, H. G. Barrow, C. M. Brown, R. M. Burstall, and R. J. Popplestone. A versatile system for computer controlled assembly. 6(2), 1975.
- [Alb81] James Albus. *Brains, Behaviour, and Robotics*. BYTE Books, Peterborough N.H., 1981.
- [AMBF83] James S. Albus, Charles R. McLean, Anthony J. Barbera, and M.L. Fitzgerald. Hierarchical control for robots in an automated factory. In *Proceedings 13th International Symposium on Industrial Robots, and Robots 7*, volume 2, Dearborn Michigan, April 1983. Robotics International of SME.
- [And77] John H. Andrae. *Thinking with the Teachable Machine*. Academic Press, London, 1977.
- [And85] Peter Merrett Andrae. Justified generalization: Acquiring procedures from examples. Technical Report 834, MIT Artificial Intelligence Laboratory, January 1985.
- [AP75] A. P. Ambler and R. J. Popplestone. Inferring the positions of bodies from specified spatial relationships. 6(2), 1975.
- [APK82] A. P. Ambler, R. J. Popplestone, and K. G. Kempf. An experiment in the offline programming of robots. In *Proceedings 12th International Symposium on Industrial Robots, 6th International Conference on Industrial Robot Technology*, 1982.

- [BA84a] Michael Brady and Philip E. Agre. The mechanics mate. In *Proceedings European Conference on Artificial Intelligence*, 1984.
- [BA84b] Michael Brady and Haruo Asada. Smoothed local symmetries and their implementation. AI Memo 757, MIT AI Lab, February 1984.
- [Bal84] D. H. Ballard. Task frames in robot manipulation. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1984.
- [Bar84] H. G. Barrow. VERIFY: A program for proving correctness of digital hardware designs. In D. G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. Elsevier Science Publishers, 1984.
- [BB83] Alan Bundy and Lawrence Byrd. Using the method of fibres in Mecho to calculate radii of gyration. In Dedre Gentner and Albert L. Stevens, editors, *Mental Models*. Lawrence Erlbaum Associates, 1983.
- [BB87] Wilhelm Burger and Bir Bhanu. Qualitative motion understanding. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.
- [BC72] H. G. Barrow and G. F. Crawford. *The Mark 1.5 Edinburgh Robot Facility*, volume 7 of *Machine Intelligence*, chapter 25, pages 465–480. Edinburgh University Press, 1972.
- [BCF86] Rodney A. Brooks, Jon Connell, and Anita Flynn. A mobile robot with onboard parallel processor and large workspace arm. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [BDdRP83] R. Bardelli, P. Dario, D. de Rossi, and P.C. Pinotti. Piezo and pyroelectric polymers skin-like tactile sensors for robots and prostheses. In *Proceedings 13th International Symposium on Industrial Robots, and Robots 7*, volume 2, Dearborn Michigan, April 1983. Robotics International of SME.

- [Ben83] Harry L. Benjamin. The development of a production robot tactile position sensor. In *Proceedings 13th International Symposium on Industrial Robots, and Robots 7*, volume 2. Robotics International of SME, 1983.
- [BF84] David A. Bourne and Mark S. Fox. Autonomous manufacturing: Automating the job shop. *IEEE Computer*, September 1984.
- [BG82] John W. Boyse and Jack E. Gilchrist. GMSolid: Interactive modeling for design and analysis of solids. *IEEE Computer Graphics and Applications*, 2(2), March 1982.
- [BH82] G. Bancon and B. Huber. Depression and dual grippers with their possible applications. In *Proceedings 12th International Symposium on Industrial Robots, 6th International Conference on Industrial Robot Technology*, 1982.
- [BH86] Robert C. Bolles and Patrice Horaud. 3DPO: a three dimensional part orientation system. *International Journal of Robotics Research*, 5(3), 1986.
- [Big86] Anton Bigelmaier. Profile of a geometrical knowledge base for CAD systems. *Computers and Graphics*, 10(4):297–306, 1986.
- [Bla86] Alan Blackwell. Artificial intelligence and New Zealand manufacturing industry. In *Proceedings of First N.Z. Conference on Robotics and Handling Automation (ROBHANZ 86)*, November 1986.
- [Bla88] Alan Blackwell. *Qualitative Geometric Reasoning Using a Partial Distance Ordering*, pages 217–229. Artificial Intelligence Developments and Applications. North-Holland, Amsterdam, 1988.
- [Bli87] A. Peter Blicher. A shape representation based on geometric topology: Bumps, gaussian curvature, and the topological zodiac. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.
- [Bob84a] D. G. Bobrow, editor. *Qualitative Reasoning about Physical Systems*. Elsevier Science Publishers, 1984.

- [Bob84b] D. G. Bobrow. Qualitative reasoning about physical systems: An introduction. In D. G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. Elsevier Science Publishers, 1984.
- [BPYA85] Michael Brady, Jean Ponce, Alan Yuille, and Haruo Asada. Describing surfaces. AI Memo 822, MIT AI Lab, January 1985.
- [Bra84] Michael Brady. *Intelligent Robots: Connecting Perception to Action*, chapter 14. The AI Business: The Commercial Uses of Artificial Intelligence. MIT Press, 1984.
- [Bra85a] Michael Brady. Artificial intelligence and robotics. 26:79–121, 1985.
- [Bra85b] David J. Braunegg. Function from form. Technical report, MIT AI Lab, November 1985.
- [Bro81] Rodney A. Brooks. Symbolic reasoning among 3D models and 2D images. 17:285–348, 1981.
- [Bro82a] Rodney A. Brooks. Symbolic error analysis and robot planning. AI Memo 685, MIT AI Lab, September 1982.
- [Bro82b] Christopher M. Brown. PADL-2: A technical summary. *IEEE Computer Graphics and Applications*, 2(2), March 1982.
- [Bro83] Rodney A. Brooks. Planning collision free motions for pick and place operations. AI Memo 725, MIT AI Lab, May 1983.
- [Bro85a] Rodney A. Brooks. A mobile robot project. Working Paper 265, MIT AI Lab, February 1985.
- [Bro85b] Rodney A. Brooks. A robust layered control system for a mobile robot. AI Memo 864, MIT AI Lab, September 1985. Also published in *IEEE Robotics and Automation*, Vol. 2, Number 1, March 1986.
- [Bro86] Rodney A. Brooks. Achieving artificial intelligence through building robots. AI Memo 899, MIT AI Lab, May 1986.
- [BRS78] M. Briot, M. Renaud, and Z. Stojiljkovic. An approach to spatial pattern recognition of solid objects. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(9), September 1978.

- [Bun78] Alan Bundy. Will it reach the top? Prediction in the mechanics world. 10(2):129–146, April 1978.
- [Byl88] Tom Bylander. A critique of qualitative simulation from a consolidation viewpoint. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(2), April 1988.
- [Cal87] Philippe Caloud. Towards continuous process supervision. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.
- [Car83] Jaime G. Carbonell. *Learning by Analogy: Formulating and Generalising Plans from Past Experience*, volume 1 of *Machine Learning*, pages 163–190. Tioga Publishing, 1983.
- [Cas87] Malcolm S. Casale. Free-form solid modeling with trimmed surface patches. *IEEE Computer Graphics and Applications*, 7(1), January 1987.
- [CB85a] Jonathan H. Connell and Michael Brady. Generating and generalizing models of visual objects. AI Memo 823, MIT AI Lab, July 1985.
- [CB85b] Jonathan H. Connell and Michael Brady. Learning shape descriptions. In *Proceedings International Joint Conference on Artificial Intelligence*, 1985.
- [CB87] Jonathon H. Connell and Michael Brady. Generating and generalizing models of visual objects. 31(1):159–183, February 1987.
- [CF87] John W. Collins and Kenneth D. Forbus. Reasoning about fluids via molecular collections. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, pages 590–594, 1987.
- [Con85] Jonathan H. Connell. Learning shape descriptions: Generating and generalizing models of visual objects. Technical Report 853, MIT AI Lab, September 1985.
- [Con87] Jonathan H. Connell. Creature design with the subsumption architecture. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.

- [Cro82] A. G. Cronshaw. Automatic chocolate decoration by robot vision. In *Proceedings 12th International Symposium on Industrial Robots, 6th International Conference on Industrial Robot Technology*, 1982.
- [Cro87] James L. Crowley. Coordination of action and perception in a surveillance robot. *IEEE Expert*, 2(4), 1987.
- [CW88] Kai-Hsiung Chang and William G Wee. A knowledge-based planning system for mechanical assembly using robots. *IEEE Expert*, March 1988.
- [D'A87] Bruce D'Ambrosio. Extending the mathematics in qualitative process theory. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, pages 595–599, 1987.
- [DAD86] Richard J. Doyle, David J. Atkinson, and Rajkumar S. Doshi. Generating perception requests and expectations to verify the execution of plans. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [Dav84a] Ernest Davis. Shape and function of solid objects: Some examples. Technical Report 137, New York University, October 1984.
- [Dav84b] Randall Davis. Diagnostic reasoning based on structure and behaviour. In D. G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. Elsevier Science Publishers, 1984.
- [Dav85] Ernest Davis. An ontology of physical actions. Technical report, MIT AI Lab, 1985.
- [DdR85] P. Dario and D. de Rossi. Tactile sensors and the gripping challenge. *IEEE Spectrum*, August 1985.
- [Dix86] John R. Dixon. Artificial intelligence and design: A mechanical engineering view. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [dK77] Johan de Kleer. Multiple representations of knowledge in a mechanics problem solver. In *Proceedings International Joint Conference on Artificial Intelligence*, pages 299–304, 1977.

- [dK79] Johan de Kleer. Qualitative and quantitative reasoning in classical mechanics. In P H Winston and R H Brown, editors, *Artificial Intelligence: An MIT Perspective*. MIT Press, 1979.
- [dK84] Johan de Kleer. How circuits work. In D. G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. Elsevier Science Publishers, 1984.
- [dKB80] Johan de Kleer and John Seely Brown. Mental models of physical mechanisms and their acquisition. In *Cognitive Skills and Their Acquisition*. Erlbaum, 1980.
- [dKB82] Johan de Kleer and John Seely Brown. Foundations of envisioning. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1982.
- [dKB83] Johan de Kleer and John Seely Brown. Assumptions and ambiguities in mechanistic mental models. In Dedre Gentner and Albert L. Stevens, editors, *Mental Models*. Lawrence Erlbaum Associates, 1983.
- [dKB84] Johan de Kleer and John Seely Brown. A qualitative physics based on confluences. In D. G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. Elsevier Science Publishers, 1984.
- [dKB86] Johan de Kleer and John Seely Brown. Theories of causal ordering. 29(1):33–62, July 1986.
- [DL84] Bruno Dufay and Jean-Claude Latombe. An approach to automatic robot programming based on inductive learning. *International Journal of Robotics Research*, 3(4), 1984.
- [dM87] Joseph di Martino. On multi-level machines for continuous speech recognition. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.
- [dMS86] Luiz S. Homem de Mello and Arthur C. Sanderson. And/or graph representation of assembly plans. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.

- [Don86] Bruce R. Donald. A theory of error detection and recovery: Robot motion planning with uncertainty in the geometric models of the robot and environment. Although unpublished when I saw it, this should since be listed in the MIT AI lab. publications list, June 1986.
- [Don87] Bruce R. Donald. A search algorithm for motion planning with six degrees of freedom. 31(3):295–353, March 1987.
- [DRD87] Philippe Dague, Olivier Raiman, and Philippe Deves. Troubleshooting: when modeling is the trouble. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, pages 600–605, 1987.
- [Ela86] Taha I. Elareef. Flavour system and message passing as representation of knowledge for solid modelling in CAD expert system. *Computers and Graphics*, 10(4):351–358, 1986.
- [ELP86] Michael Erdmann and Tomas Lozano-Perez. On multiple moving objects. AI Memo 883, MIT AI Lab, May 1986.
- [EUY⁺72] Masakazu Ejiri, Takeshi Uno, Hauto Yoda, Tatsuo Goto, and Kiyoo Takeyasu. A prototype intelligent robot that assembles objects from plan drawings. *IEEE Transactions on Computers*, 21(2), February 1972.
- [FA86] W. Friedrich and G. Arndt. Vision-aided flexible component handling. In *Proceedings of First N.Z. Conference on Robotics and Handling Automation (ROBHANZ 86)*, November 1986.
- [Fah73] Scott E. Fahlman. A planning system for robot construction tasks. Technical Report 283, MIT AI Lab, May 1973.
- [Fal86] Boi Faltings. A theory of qualitative kinematics in mechanisms. Research report, University of Illinois, May 1986.
- [Fal87] Boi Faltings. A theory of qualitative kinematics in mechanisms. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.
- [FFG86] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The structure-mapping engine. In *Proceedings of the National*

- Conference of the American Association for Artificial Intelligence*, 1986.
- [FG86] Kenneth D. Forbus and Dedre Gentner. Causal reasoning about quantities. In *Proc. of the eighth annual meeting of the Cognitive Science Society*, August 1986.
- [FH84] Ronald S. Fearing and John M. Hollerbach. Basic solid mechanics for tactile sensing. AI Memo 771, MIT AI Lab, March 1984.
- [FH86] O. D. Faugeras and M. Hebert. The representation, recognition, and locating of 3D objects. *International Journal of Robotics Research*, 5(3), 1986.
- [Fin84] Aryeh Finegold. *The Engineer's Apprentice*, chapter 9. The AI Business: The Commercial Uses of Artificial Intelligence. MIT Press, 1984.
- [Fle85a] Margaret Morrison Fleck. Local rotational symmetries. Technical Report 852, MIT AI Lab, August 1985.
- [Fle85b] Alan Fleming. Analysis of uncertainties in a structure of parts. In *Proceedings International Joint Conference on Artificial Intelligence*, 1985.
- [FN71] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. 2:189–208, 1971.
- [FNF87] Kenneth D. Forbus, Paul Nielsen, and Boi Faltings. Qualitative kinematics: A framework. Technical Report UIUCDCS-R-87-1352, Dept. of Computer Science, University of Illinois at Urbana-Champaign, June 1987.
- [For81] Kenneth D. Forbus. A study of qualitative and geometric knowledge in reasoning about motion. Technical Report 615, MIT AI Lab., February 1981.
- [For82] Kenneth D. Forbus. Modeling motion with qualitative process theory. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1982.

- [For83] Kenneth D. Forbus. Qualitative reasoning about space and motion. In Dedre Gentner and Albert L. Stevens, editors, *Mental Models*. Lawrence Erlbaum Associates, 1983.
- [For84a] Kenneth D. Forbus. Qualitative process theory. Technical Report 789, MIT AI Lab, July 1984.
- [For84b] Kenneth D. Forbus. Qualitative process theory. In D. G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. Elsevier Science Publishers, 1984.
- [For86] Kenneth D. Forbus. Interpreting measurements of physical systems. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [For87] Kenneth D. Forbus. The logic of occurrence. In *Proceedings International Joint Conference on Artificial Intelligence*, pages 409–415, 1987.
- [FT87] B. Faverjon and P. Tournassoud. The mixed approach for motion planning: Learning global strategies from a local planner. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.
- [Fun80] B. V. Funt. Problem-solving with diagrammatic representations. 13(3):201–230, May 1980.
- [GB75] D. D. Grossman and M. W. Blasgen. Orienting mechanical parts by computer controlled manipulator. *IEEE Transactions on Systems, Man, and Cybernetics*, 5(5), September 1975.
- [GB87] Douglas S. Green and David C. Brown. Qualitative reasoning during design about shape and fit: A preliminary report. In *Expert Systems in Computer-Aided Design*. IFIP, 1987.
- [GDG⁺85] Maria Gini, Rajkumar Doshi, Marc Gluch, Richard Smith, and Imran Zualkernan. The role of knowledge in the architecture of a robust robot control. In *Proceedings IEEE Conference on Robotics and Automation*, 1985.
- [Gel63] H. Gelernter. Realization of a geometry-theorem proving machine. In Edward A. Feigenbaum and Julian Feldman, editors, *Computers and Thought*, pages 134–152. McGraw-Hill, 1963.

- [Gen84] M. R. Genesereth. The use of design descriptions in automated diagnosis. In D. G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. Elsevier Science Publishers, 1984.
- [GHL63] H. Gelernter, J. R. Hansen, and D. W. Loveland. Empirical explorations of the geometry-theorem proving machine. In Edward A. Feigenbaum and Julian Feldman, editors, *Computers and Thought*, pages 153–163. McGraw-Hill, 1963.
- [Gou86] Laurent Gouzenes. Strategies for solving collision-free trajectories problems for mobile and manipulator robots. *International Journal of Robotics Research*, 4(4), 1986.
- [Gre83] James G. Greeno. Conceptual entities. In Dedre Gentner and Albert L. Stevens, editors, *Mental Models*. Lawrence Erlbaum Associates, 1983.
- [Gro76] D. D. Grossman. Procedural representation of three-dimensional objects. *IBM Journal of Research and Development*, 20(6), November 1976.
- [GSCT83] William A. Gruver, Barry I. Soroka, John J. Craig, and Timothy L. Turner. Evaluation of commercially available robot programming languages. In *Proceedings 13th International Symposium on Industrial Robots, and Robots 7*, volume 2, Dearborn Michigan, April 1983. Robotics International of SME.
- [GT78] D. D. Grossman and R. H. Taylor. Interactive generation of object models with a manipulator. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(9), September 1978.
- [GTI80] T. Goto, K. Takeyasu, and T. Inoyama. Control algorithm for precision insert operation robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(1), January 1980.
- [GZS88] David C. Gossard, Robert P. Zuffante, and Hiroshi Sakurai. Representing dimensions, tolerances, and features in MCAE systems. *IEEE Expert*, March 1988.
- [HA85] John M. Hollerbach and Christopher G. Atkeson. Characterization of joint-interpolated arm movements. AI Memo 849, MIT AI Lab, June 1985.

- [HA87] Patrick J. Hayes and James F. Allen. Short time periods. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.
- [Hay78] Patrick J. Hayes. The naive physics manifesto. In D. Michie, editor, *Expert Systems in the Micro-Electronic Age*. Edinburgh University Press, 1978.
- [Hay83] Patrick J. Hayes. The second naive physics manifesto. Cognitive Science Technical Report URCS-10, University of Rochester, October 1983.
- [Hil82] Robin Hillyard. The build group of solid modelers. *IEEE Computer Graphics and Applications*, 2(2), March 1982.
- [Hol79] John M. Hollerbach. Understanding manipulator control by synthesising human handwriting. In P H Winston and R H Brown, editors, *Artificial Intelligence: An MIT Perspective*. MIT Press, 1979.
- [Hor79] B. K. P. Horn. Kinematics, statics, and dynamics of two dimensional manipulators. In P H Winston and R H Brown, editors, *Artificial Intelligence: An MIT Perspective*. MIT Press, 1979.
- [HR86] Mark L. Hornick and Bahram Ravani. Computer-aided off-line planning and programming of robot motion. *International Journal of Robotics Research*, 4(4), 1986.
- [HSS84] J. E. Hopcroft, J. T. Schwartz, and M Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the “Warehouseman’s Problem”. *International Journal of Robotics Research*, 3(4), 1984.
- [HW86] John Hopcroft and Gordon Wilfong. Motion of objects in contact. *International Journal of Robotics Research*, 4(4), 1986.
- [Ino74] H. Inoue. Force feedback in precise assembly tasks. AI Memo 308, MIT AI Lab, August 1974.
- [Ino79] H. Inoue. Force feedback in precise assembly tasks. In P H Winston and R H Brown, editors, *Artificial Intelligence: An MIT Perspective*. MIT Press, 1979.

- [Ino85] H. Inoue. Building a bridge between AI and robotics. In *Proceedings International Joint Conference on Artificial Intelligence*, 1985.
- [IS85] Jehuda Ish-Shalom. The CS language concept: A new approach to robot motion design. *International Journal of Robotics Research*, 4(1), 1985.
- [IS86a] Y. Iwasaki and H. A. Simon. Causality in device behaviour. 29(1):3–32, July 1986.
- [IS86b] Y. Iwasaki and H. A. Simon. Theories of causal ordering: Reply to de Kleer and Brown. 29(1):63–72, July 1986.
- [ISKM87] Masaru Ishii, Shigeyuki Sakane, Masayoshi Kakikura, and Yoshio Mikami. A 3-D sensor system for teaching robot paths and environments. *International Journal of Robotics Research*, 6(2), 1987.
- [JB86] R. A. Jarvis and J. C. Byrne. Robot navigation: Touching, seeing and knowing. In *Proceedings of 1st Australian Artificial Intelligence Conference*, 1986.
- [Jos87] Leo Joskowicz. Shape and function in mechanical devices. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, pages 611–615, 1987.
- [JWKB84] S. C. Jacobsen, J. E. Wood, D. F. Knutti, and K. B. Biggers. The UTAH/M.I.T. dextrous hand: Work in progress. *International Journal of Robotics Research*, 3(4), 1984.
- [KA86] Y. C. Kim and J. K. Aggarwal. Rectangular parallelepiped coding: A volumetric representation of three-dimensional objects. *IEEE Robotics and Automation*, 2(3), 1986.
- [KBC⁺86] A. C. Kak, K. L. Boyer, C. H. Chen, R.J. Safranek, and H. S. Yang. A knowledge-based robotic assembly cell. *IEEE Expert*, 1(2):63–85, 1986.
- [KC87] Benjamin J. Kuipers and Charles Chiu. Taming intractible branching in qualitative simulation. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.

- [KD86] S. K. Kambhampati and L. S. Davis. Multiresolution path planning for mobile robots. *IEEE Robotics and Automation*, 2(3), 1986.
- [Kem85] K. G. Kempf. Manufacturing and artificial intelligence. *Robotics*, 1(1), 1985.
- [Kha85] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings IEEE Conference on Robotics and Automation*, 1985.
- [KJ86] Thomas F. Knoll and Ramesh C. Jain. Recognizing partially visible objects using feature indexed hypotheses. *IEEE Robotics and Automation*, 2(1), March 1986.
- [KL88] Benjamin J. Kuipers and Tod S. Levitt. Navigation and mapping in large-scale space. *AI Magazine*, 9(2), 1988.
- [KMMN85] D. Kapur, J. Mundy, D. Musser, and P. Narendran. Reasoning about three dimensional space. In *Proceedings IEEE Conference on Robotics and Automation*, 1985.
- [Kna85] T. M. Knasel. Fifth international conference on assembly automation. *Robotics*, 1(1), 1985.
- [KTTP83] D.R. Kemp, G.E. Taylor, P.M. Taylor, and A. Pugh. A sensory gripper for handling textiles. In *Proceedings 13th International Symposium on Industrial Robots, and Robots 7*, volume 2, Dearborn Michigan, April 1983. Robotics International of SME.
- [Kui82] Benjamin J. Kuipers. Getting the envisionment right. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1982.
- [Kui84] Benjamin J. Kuipers. Commonsense reasoning about causality: Deriving behaviour from structure. In D. G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. Elsevier Science Publishers, 1984.
- [Kui86] Benjamin J. Kuipers. Qualitative simulation. 29(3):289–338, September 1986.

- [LB83] D. B. Lenat and J. S. Brown. Why AM and Eurisko appear to work. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1983.
- [Len83] D. B. Lenat. *The Role of Heuristics in Learning by Discovery: Three Case Studies*, volume 1 of *Machine Learning*, pages 243–306. Tioga Publishing, 1983.
- [LF87] Y.C. Lee and K.S. Fu. Machine understanding of CSG: Extraction and unification of manufacturing features. *IEEE Computer Graphics and Applications*, 7(1), January 1987.
- [LK65] J. Y. S. Luh and R. J. Krolak. A mathematical model for mechanical part description. *Communications of the ACM*, 8(2), February 1965.
- [Low87] David G. Lowe. Three-dimensional object recognition from single two-dimensional images. 31(3):355–395, March 1987.
- [LP76] Tomas Lozano-Perez. The design of a mechanical assembly system. Technical Report 397, MIT AI Lab, December 1976.
- [LP79] Tomas Lozano-Perez. A language for automated mechanical assembly. In P H Winston and R H Brown, editors, *Artificial Intelligence: An MIT Perspective*. MIT Press, 1979.
- [LP82] Tomas Lozano-Perez. Robot programming. AI Memo 698a, MIT AI Lab, December 1982.
- [LP83] Tomas Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 32(2), February 1983.
- [LP85] Tomas Lozano-Perez. Compliance in robot manipulation. 25(1), January 1985.
- [LP88] Mark Lee and Chris Price. Towards deeper systems: When will they ever understand? In *Proceedings of the New Zealand Conference on Expert Systems*, May 1988.
- [LPB85] Tomas Lozano-Perez and Rodney A. Brooks. An approach to automatic robot programming. AI Memo 842, MIT AI Lab, April 1985.

- [LPD83] Du Lian, Steven Peterson, and Max Donath. A three-fingered, articulated, robotic hand. In *Proceedings 13th International Symposium on Industrial Robots, and Robots 7*, volume 2, Dearborn Michigan, April 1983. Robotics International of SME.
- [LPMT83] Tomas Lozano-Perez, Matthew T. Mason, and Russell H. Taylor. Automatic synthesis of fine-motion strategies for robots. AI Memo 759, MIT AI Lab, December 1983.
- [LPW79] Tomas Lozano-Perez and M. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22:560–570, 1979.
- [LS83] A. Levas and M. Selfridge. Voice communications with robots. In *Proceedings 13th International Symposium on Industrial Robots, and Robots 7*, volume 2, Dearborn Michigan, April 1983. Robotics International of SME.
- [LW77] L. I. Lieberman and M. A. Wesley. Autopass: an automatic programming system for computer controlled mechanical assembly. *IBM Journal of Research and Development*, 21(4), July 1977.
- [Lyo85] Damian M. Lyons. A simple set of grasps for a dextrous hand. In *Proceedings IEEE Conference on Robotics and Automation*, 1985.
- [MA87] Seshashayee S. Murthy and Sanjaya Addaanki. PROMPT: an innovative design tool. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, pages 637–642, 1987.
- [Mac84] B. A. MacDonald. *Designing Teachable Robots*. PhD thesis, University of Canterbury, 1984.
- [Mac87a] B. A. MacDonald. Improved robot design. *Transactions of the Institution of Professional Engineers New Zealand*, 14(1/EMCh), March 1987.
- [Mac87b] P. Mackerras. Formulating and testing simple boundary hypotheses in textured images. In *Proceedings Australian Joint Conference on Artificial Intelligence (AI'87)*, November 1987.

- [Mas81] Matthew T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(6), June 1981.
- [Mas85] Matthew T. Mason. The mechanics of manipulation. In *Proceedings IEEE Conference on Robotics and Automation*, 1985.
- [Mas86] Matthew T. Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research*, 5(3), 1986.
- [MB86] Raymond J. Mooney and Scott W. Bennett. A domain independent explanation-based generalizer. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [McC79] Pamela McCorduck. *Machines that Think*. W. H. Freeman and Company, 1979.
- [McC83] Michael McCloskey. Naive theories of motion. In Dedre Gentner and Albert L. Stevens, editors, *Mental Models*. Lawrence Erlbaum Associates, 1983.
- [Mey81] Jeanine Meyer. An emulation system for programmable sensory robots. *IBM Journal of Research and Development*, 25(6), November 1981.
- [MF87] Sanjay Mittal and Felix Frayman. Making partial choices in constraint reasoning systems. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, pages 631–636, 1987.
- [MHCM86] R. L. Madarasz, L. C. Heiny, R. F. Crompt, and N. M. Mazur. The design of an autonomous vehicle for the disabled. *IEEE Robotics and Automation*, 2(3), 1986.
- [MS87] Michael L. Mavrouniotis and George Stephanopoulos. Reasoning with orders of magnitude and approximate relations. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, pages 626–630, 1987.

- [MUB83] Tom M. Mitchell, Paul E. Utgoff, and Ranan Banerji. *Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics*, volume 1 of *Machine Learning*, pages 163–190. Tioga Publishing, 1983.
- [Ngu85a] Van-Duc Nguyen. The synthesis of force-closure grasps in the plane. AI Memo 861, MIT AI Lab, September 1985.
- [Ngu85b] Van-Duc Nguyen. The synthesis of stable grasps in the plane. AI Memo 862, MIT AI Lab, October 1985.
- [NR76] David Nitzan and Charles Rosen. Programmable industrial automation. *IEEE Transactions on Computers*, 25(12), December 1976.
- [PAB80] R. J. Popplestone, A. P. Ambler, and I. M. Bellos. An interpreter for a language for describing assemblies. 14(1), 1980.
- [PCR86] Tsaiyun Phillips, Robert Cannon, and Azriel Rosenfeld. Decomposition and approximation of three-dimensional solids. *Computer Vision, Graphics, and Image Processing*, 33(3):307–317, March 1986.
- [Pen86a] Alex P. Pentland. Parts: Structured descriptions of shape. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [Pen86b] Alex P. Pentland. Perceptual organisation and the representation of natural form. 28(3):293–331, May 1986.
- [Pen87] Alex P. Pentland. A new sense for depth of field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4), July 1987.
- [Pli85] Philip I. Plimmer. Common sense reasoning about physical devices. Victoria University of Wellington Research Project Report, October 1985.
- [Pop79] R. J. Popplestone. Relational programming. In J.E. Hayes, D. Michie, and L.I. Mikulich, editors, *Machine Intelligence 9*, chapter 7. Ellis Horwood, 1979.

- [Pug82] Alan Pugh. Second generation robotics. In *Proceedings 12th International Symposium on Industrial Robots, 6th International Conference on Industrial Robot Technology*, 1982.
- [Rai86] Olivier Raiman. Order of magnitude reasoning. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [RC86] A. A. G. Requicha and S. C. Chan. Representation of geometric features, tolerances, and attributes in solid modelers based on constructive geometry. *IEEE Robotics and Automation*, 2(3), 1986.
- [RJ88] A. Ravishankar Rao and Ramesh Jain. Knowledge representation and control in computer vision systems. *IEEE Expert*, March 1988.
- [RKH85] Whitman Richards, Jan J. Koenderink, and D. D. Hoffman. Inferring 3D shapes from 2D codons. AI Memo 840, MIT AI Lab, April 1985.
- [RS88] Gudula Retz-Schmidt. Various views on spatial prepositions. *AI Magazine*, 9(2), 1988.
- [RT82] M. H. Raibert and J. E. Tanner. A VLSI tactile array sensor. In *Proceedings 12th International Symposium on Industrial Robots, 6th International Conference on Industrial Robot Technology*, 1982.
- [Rus84] Paul M. Russo. *Intelligent Robots: Myth or Reality*, chapter 16. The AI Business: The Commercial Uses of Artificial Intelligence. MIT Press, 1984.
- [RV82] A. A. G. Requicha and H. B. Voelcker. Solid modeling: A historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, 2(2), March 1982.
- [Sac87] Elisha Sacks. Hierarchical reasoning about inequalities. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, pages 649–654, 1987.

- [SB87] Sargur N. Srihari and Radmilo M. Bozinovic. A multi-level perception approach to reading cursive script. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.
- [SF86] Devika Subramanian and Joan Feigenbaum. Factorization in experiment generation. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [Shm86] James G. Shmolze. Physics for robots. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [Sie86] David Mark Siegel. Contact sensors for dextrous robotic hands. Technical Report 900, MIT AI Lab, June 1986.
- [Sim82] Reid Simmons. Spatial and temporal reasoning in geologic map interpretation. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1982.
- [Sim86] Reid Simmons. “Commonsense” arithmetic reasoning. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [SJ85] Alberto Maria Segre and Gerald De Jong. Explanation-based manipulator learning: Acquisition of planning ability through observation. In *Proceedings IEEE Conference on Robotics and Automation*, pages 555–560, 1985.
- [SL83] M. Selfridge and A. Levas. Teaching robots by example. In *Proceedings 13th International Symposium on Industrial Robots, and Robots 7*, volume 2, Dearborn Michigan, April 1983. Robotics International of SME.
- [SLK86] Michael O. Shneier, Ronald Lumia, and Ernest W. Kent. Model-based strategies for high-level robot vision. *Computer Vision, Graphics, and Image Processing*, 33(3):293–306, March 1986.
- [SM87] Marc G. Slack and David P. Miller. Path planning through time and space in dynamic domains. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.

- [SS86] A. Sharma and S. A. R. Scrivener. Constructing 3-D object models using multiple simulated 2.5-D sketches. *International Journal of Man-Machine Studies*, 24(6):633–644, June 1986.
- [SS88] Mark Segal and Carlo H. Sequin. Partitioning polyhedral objects into nonintersecting parts. *IEEE Computer Graphics and Applications*, 8(1), January 1988.
- [ST87] Sergio W. Sedas and Sarosh N. Talukdar. Disassembly planner for redesign. Internal document from Engineering Design Research Center, Carnegie Mellon University, 1987.
- [Sta83a] Craig Stanfill. The decomposition of a large domain: Reasoning about machines. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, August 1983.
- [Sta83b] Craig Stanfill. *Form and Function: The Representation of Machines*. PhD thesis, University of Maryland, 1983.
- [Sug86] Kokichi Sugihara. *Machine Interpretation of Line Drawings*. The MIT Press Series in Artificial Intelligence. MIT Press, Cambridge, Massachusetts, 1986.
- [Sus75] Gerald Jay Sussman. *A Computer Model of Skill Acquisition*. Elsevier, 1975.
- [SV86] M. Selfridge and W. Vannoy. A natural language interface to a robot assembly system. *IEEE Robotics and Automation*, 2(3), 1986.
- [Ten86] Josh Tenenbunrg. Planning with abstraction. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [TKO82] J. P. Trevylan, P. D. Kovesi, and M. C. H. Ong. Techniques for surface representation and adaptation in automated sheep shearing. In *Proceedings 12th International Symposium on Industrial Robots, 6th International Conference on Industrial Robot Technology*, 1982.

- [TO83] Masaharu Takano and Gotaro Odawara. Development of new type of mobile robot TO-ROVER. In *Proceedings 13th International Symposium on Industrial Robots, and Robots 7*, volume 2. Robotics International of SME, 1983.
- [TPB81] K. Takase, R. P. Paul, and E. J. Berg. A structured approach to robot programming and teaching. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(4), April 1981.
- [Tre79] Kenneth R. Treer. *Automated Assembly*. Society of Manufacturing Engineers, Dearborn, Michigan, 1979.
- [TST82] P. M. Taylor, K. K. W. Selke, and G. E. Taylor. Closed loop control of an industrial robot using visual feedback from a sensory gripper. In *Proceedings 12th International Symposium on Industrial Robots, 6th International Conference on Industrial Robot Technology*, 1982.
- [Vil84] Philippe Villers. *Intelligent Robots: Moving toward Megassembly*, chapter 15. The AI Business: The Commercial Uses of Artificial Intelligence. MIT Press, 1984.
- [VP86] Raul E. Valdes-Perez. Spatio-temporal reasoning and linear inequalities. AI Memo 875, MIT AI Lab, May 1986.
- [VW86] W. K. Veitschegger and C. H. Wu. Robot accuracy analysis based on kinematics. *IEEE Robotics and Automation*, 2(3), 1986.
- [WA85] Harry West and Haruhiko Asada. A method for the design of hybrid position/force controllers for manipulators constrained by contact with the environment. In *Proceedings IEEE Conference on Robotics and Automation*, 1985.
- [Wel87] Daniel S. Weld. Comparative analysis. In *Proceedings International Joint Conference on Artificial Intelligence*, 1987.
- [WF86] E. K. Wong and K. S. Fu. A hierarchical orthogonal space approach to three-dimensional path planning. *IEEE Robotics and Automation*, 2(1), March 1986.

- [WHK88] Ellen L. Walker, Martin Herman, and Takeo Kanade. A framework for representing and reasoning about three-dimensional objects for vision. *AI Magazine*, 9(2), 1988.
- [Wil84] Brian C. Williams. Qualitative analysis of MOS circuits. In D. G. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. Elsevier Science Publishers, 1984.
- [Wil86] Brian C. Williams. Doing time: Putting qualitative reasoning on firmer ground. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*, 1986.
- [WL85] M.C. Wu and C.R. Liu. Automated process planning and expert systems. In *Proceedings IEEE Conference on Robotics and Automation*. IEEE Computer Society Press, March 1985.
- [WLPL⁺80] M. A. Wesley, T. Lozano-Perez, L. I. Lieberman, M. A. Lavin, and D. D. Grossman. A geometric modelling system for automated mechanical assembly. *IBM Journal of Research and Development*, 24(1), January 1980.
- [WM81] M. A. Wesley and G. Markowsky. Fleshing out projections. *IBM Journal of Research and Development*, 25(6), November 1981.
- [Woj87] Zbigniew Wojcik. Rough approximation of shapes in pattern recognition. *Computer Vision, Graphics, and Image Processing*, 40(2), November 1987.
- [Woo84] Tony C. Woo. Interfacing solid modelling to CAD and CAM: Data structures and algorithms for decomposing a solid. *IEEE Computer*, December 1984.
- [WP84] Patrick H. Winston and Karen A. Prendergast. *From the Blocks World to the Business World*, chapter 22. The AI Business: The Commercial Uses of Artificial Intelligence. MIT Press, 1984.
- [Wri83] Allen J. Wright. End effector technology and programmed automatic exchange. In *Proceedings 13th International Symposium on Industrial Robots, and Robots 7*, volume 2, Dearborn Michigan, April 1983. Robotics International of SME.

- [xZ87] Wei xiong Zhang. Goal-level automatic robot programming – system and approach. In *Proceedings Australian Joint Conference on Artificial Intelligence (AI'87)*, November 1987.
- [Yea85] W. K. Yeap. An ASR [absolute space representation] of the environment. Technical report, Department of Computer Science, University of Otago, Dunedin, N.Z., 1985.
- [YFK84] Hiroshi Yoshiura, Kikuo Fujimura, and Tosiyasu L Kunii. Top-down construction of 3D mechanical object shapes from engineering drawings. *IEEE Computer*, December 1984.
- [Yin87] B. Yin. Using vision data in an object-level robot language – RAPT. *International Journal of Robotics Research*, 6(1), 1987.
- [Zad79] L. A. Zadeh. A theory of approximate reasoning. In J.E. Hayes, D. Michie, and L.I. Mikulich, editors, *Machine Intelligence 9*, chapter 7. Ellis Horwood, 1979.