Geometric Reasoning

R.F. Woodbury and I.J. Oppenheim
Carnegie-Mellon University
Pittsburgh, PA 15213

1. Introduction

1.1. Cognitive robotics and space telerobotization

Cognitive robot systems are ones in which sensing and representation occur, from which task plans and tactics are determined. Such a robot system accomplishes a task after being told "what" to do, but determines for itself "how" to do it. Cognition is *required* when the work environment is uncontrolled, when contingencies are prevalent, or when task complexity is large; it is *useful* in any robotic mission. A number of distinguishing features can be associated with cognitive robotics, and one to be emphasized here is the role of artificial intelligence in knowledge representation and in planning. While space *telerobotics* may elude *some* of the problems driving cognitive robotics, it shares many of the same demands, and it can be assumed that capabilities developed for cognitive robotics can be employed advantageously for telerobotics in general.

The top level problem is *task planning*, and it is appropriate to introduce a hierarchical view of control. Presented with certain mission objectives the system must generate plans (typically) at the strategic, tactical, and reflexive levels. The structure by which knowledge is used to construct and update these plans endows the system with its cognitive attributes, and with the ability to deal with contingencies, changes, unknowns, and so on. This paper is not on the topic of AI task planning *per se*, but rather on issues of representation and reasoning which are absolutely fundamental to robot manipulations decisions based upon *geometry*, are discussed here, not AI task planning per se.

1.2. Representation and Manipulation of Geometric Information

The title to this section is the essential statement of our research interest; it can be paraphrased as *geometric reasoning*. Consider the following steps in problem solving for a robotic manipulation problem:

- Identification of the environment and objects within it.
- Identification of physical relationships between objects.
- Generation of plans to accomplish robotic task objectives; decisions at different levels of abstraction.
- Execution of any manipulator motion.

Complex cognition on geometric objects is reflected in every step. The research question before us is to permit computer-based reasoning on geometry, utilizing constructs that human designers and users of robotic systems can employ. A particular purpose is the provision of an environment in which a knowledge based system (KBS) can be implemented which presents geometric knowledge employing the powerful concepts that humans use, and which does so uniformly with other knowledge in the KBS.

Geometric reasoning spans issues in the representation, and intelligent manipulation of models which describe geometric objects. It is of general applicability over the entire range of disciplines which address real-world physical objects. While differences in the information requirements of various disciplines exist, the commonalities with respect to geometric information are marked. This research takes as its point of departure these commonalities and seeks an architecture for geometric reasoning. The methodology of the research has two related aspects. The first is a search for concepts around which such an architecture may be organized. The second is the development of a prototype system, both as a medium for exploration and as a demonstration of concept.

The group of computer languages known as knowledge based expert system languages are of greatest interest, as it is in these languages that much of the current work in real-world intelligent computation is being expressed. Of particular interest are object oriented languages, for two reasons: 1) they provide convenient means for the description of and computation on inherently hierarchical information, 2) techniques are rapidly developing to integrate the other main forms of expert system programming, rules and logic programming, into the object oriented paradigm.

1.3. Outline of Paper

Section 2 discusses the *Background* to work in this area: geometric reasoning itself (where it has been identified as such), treatment of constraints, certain origins within artificial intelligence proper, geometric modelling, computational geometry, and graphics. Section 3 contains our proposed *Architecture* for geometric reasoning: the requirements, the concepts brought to our design, the system description, and a brief statement of the present implementation. Section 4 contains the *Conclusions and Recommendations*, including an outline of further approaches to geometric reasoning by some colleagues at Carnegie-Mellon University.

2. Background

Developments in several disciplines provide starting places and points of departure:

Geometric Reasoning itself has a significant, but modest and scattered, literature. Kuipers [Kuipers 77] created a a theoretical model of human spatial cognition using concepts of representation developed by Lynch [Lynch 60]. These concepts are related together in a network, through which propagation is used to perform inference. An implicit hierarchical organization is provided by using containment relations on spatial entities. Brooks [Brooks 81] developed a constraint based geometric reasoner as part of of the ACRONYM vision system. The reasoner maintains a class hierarchy (Brooks calls it a restriction graph) of geometric representations based upon generalized cylinders. The parameters which determine a generalized cylinder are restricted through the use of algebraic constraints. Geometric reasoning is accomplished by algebraic and numerical methods to determine bounds on satisfiability of constraint sets and extremum of objective functions. McDermott described metric spatial inference, a technique to make inferences about the relative locations of objects based on simple descriptions. This work was extended by Davis [Davis 81]. Constraints on the relative locations of coordinate systems are used to define *fuzzy maps* which capture the constraint information in a form amenable to computation of queries. Davis [Davis 86] greatly expanded on this earlier work to develop a theory of cognitive mapping which particularly addressed issues of representation, retrieval and assimilation. The seminal contribution of this work is the formalization of a means of building a map incrementally and of performing inferences on incomplete maps. Akiner [Akiner 85] built a geometric reasoner based on resolution theorem proving in the domain of right rectangular orthogonally oriented cuboids. Geometric objects are expressed as assertions in Prolog and reasoning is accomplished by application of the backtracking search in Prolog using a set of rules about geometry. Dixon [Libardi 86] [Nielson 86] has used the concept of features to build various specialized representations for design problems. Features are groupings of boundary elements of a solid that provide convenient units of concept to application programs. Wing and Arbab [Wing 85] outlined a deductive geometric reasoning system. The main contribution of their proposal is recognition of the need for a clear language for geometric reasoning.

Constraints are the basis for certain types of computation which seek to maintain some characteristics of an object constant under modification of the object. Constraints are intimately related to dimensioning, tolerancing and variational geometry. The literature on constraints is large and growing. Sutherland [Sutherland 63] created SKETCHPAD, recognized as the seminal work in the area. The contributions of SKETCHPAD include propagation[1], numerical solution through relaxation and a method (pins) for linking constrained objects. Steele and Sussman [Sussman 80] present a language for the representation of almost hierarchical constraint networks, a method of explanation of constraint calculations and a simple solution technique called *local propagation*. They also discuss algebraic transformations of constraint networks. Steele, in his dissertation [Steele 80], deeply examines the implementation of constraint systems. Borning [Borning 79] developed a programming language dominated by constraints. His contributions included a strong use of the object oriented programming paradigm, a local and fast propagation algorithm and the use of planning for propagation. Light and Gossard [Light 82], demonstrate *variational geometry*, an instance of the relaxation algorithm used to dynamically manipulate parametric primitives. Gosling [Gosling 83] contributed several new techniques for constraint satisfaction. These include: the use of breadth-first search to plan propagation, the use of automatic transformation of constraint networks and a fast graph isomorphism algorithm to make frequent use of transformation feasible.

Artificial Intelligence and related fields provide key concepts of search, hierarchical knowledge representation, inheritance, theorem proving and deduction. Specific results in robot task planning [Fikes 71] [Fahlman 74], geologic map interpretation [Simmons 82] and real-world simulation [Klahr 82] provide useful precedents. STRIPS [Fikes 71] demonstrated robot task planning in a circumscribed domain. In contrast, Fahlman [Fahlman 74] developed a planning system which was highly dependent on a powerful underlying model. The resulting simplification of the actual planner demonstrates the effectiveness of isolating geometric issues. Simmons [Simmons 82] used a diagramming scheme separate from the rest of his program to compute adjacencies, positions and orientation of parts of geologic formations. Klahr [Klahr 82] discussed various approaches to geometric relationships in the development of an object-oriented battle simulator. His conclusions concerning the use of so-called *auxiliary objects* strongly support an independent treatment of geometric information.

Geometric Modelling grew out of early applications of computers to design and manufacture [Requicha 80] [Eastman 79]. Geometric Modelling is concerned with the representation and manipulation of subsets of three-dimensional Euclidean space and with the construction of computer systems to support these tasks. Several significant and distinct methods of representation are derived from Geometric Modelling. These methods are: pure primitive instancing, spatial occupancy enumeration, tree decomposition, sweep representation, cell decomposition, constructive solid geometry and boundary representation. Of these, tree decomposition, constructive solid geometry and boundary representation are of most significance to this work. In recent years a great number of results have emerged in this field.

Computational Geometry is, according to Preparata and Shamos [Preparata 85] an attempt to *'reshape - whenever necessary - the*

[1] Called the one pass method by Sutherland.

classical discipline (of geometry) into its computational incarnation." This field is concerned with the design and analysis of problems in the construction and computation of properties of geometric objects. Preparata classifies problems in computational geometry into five categories, each describing problems in one of the following areas: convex hull, intersection, geometric search, proximity and geometric optimization. From this field may be garnered good algorithms for these problems.

Computer Graphics has spawned a wide variety of techniques for the creation of images on computer output devices and for visually based human-machine interaction. The fundamentals of this field, especially rendering algorithms and interaction techniques have relevance to the present research and are of interest for the services they can provide in creating a usable geometric tool.

3. An Architecture for Geometric Reasoning

3.1. Requirements

An architecture for geometric reasoning must meet several requirements. It must:

- Represent a wide **DOMAIN OF GEOMETRIC ENTITIES.** A geometric entity is a subset of three dimensional Euclidean space (R^3). Combinations of restrictions based upon finite describability, compactness and regularity can be used to specify a large and useful set of classes of geometric entities. A myriad of algorithms exist to represent and manipulate members of these classes.
- Represent a wide **DOMAIN OF SPATIAL RELATIONSHIPS.** An object in space is located with reference to some other object or to some coordinate frame. Such a reference establishes a relation between the located object and its referent. Relationships are a key component of modelling systems as most modelled objects are not homogeneous but consist of many interrelated parts. Many types of relationships exist; of particular interest here kinematic, locational and topological relationships.
- Support a wide variety of useful **QUERIES** on geometric data. Queries take as input a description of subsets of R^3 and produce as results another subset of R^3, a vector, a scalar or some textual description. The results of queries may be considered as partial models of the original subset of R^3.
- Support the general **CLASSIFICATION** of geometric objects. Geometric entities may be classified by the results of any query on those entities. A classification scheme examines a representation of geometric information and determines if it denotes an object which belongs to any of the classes defined in the scheme.
- Provide a general means to **GENERATE** families of objects. An incomplete model of a geometric object defines a class of objects, C, all of which are partially described by the incomplete model, but which differ in characteristics not described by the partial model[2]. A generation algorithm produces members of the class C when given as input an incomplete geometric model. If the generation algorithm can produce all such members, it is said to be exhaustive.
- Provide a wide variety of **MODIFICATION TECHNIQUES** on geometric data. A modification of geometric data is an operator which changes some aspect of the data while leaving other portions unaffected. Both object definition and object altering operators are modification techniques.
- Be directly **ACCESSIBLE** from a knowledge based system. A geometric reasoner is intended to be used as a utility for the representation and manipulation of geometric information within a knowledge based system. It must be accessible from such a system in a clear, well-defined manner.
- Be **EXTENSIBLE** to include new algorithms for geometric computation, without structural change to the system. Both the number of applications which use geometry and the number of algorithms on geometric representations are growing explosively. A geometric reasoner must be able to gracefully accept the addition of new algorithms and representations if it is to continue to be useful.

3.2. Concepts

The requirements above pose constraints on the performance of a system, not upon the means of achieving such performance. We have developed a system architecture within which these requirements may be achieved. This architecture is comprised of three component concepts: classes of spatial sets, geometric abstractions and features. The following describes each of these concepts individually and then relates the concepts together to form the overall architecture.

3.2.1. Classes of spatial sets

Spatial sets may be organized into classes. A class is distinguished by certain properties that must be true for all of its members. For example, the class of regular polyhedra consists of those polyhedra which have identical vertices and dihedral angles and faces which are congruent, convex and regular. Classes are recursive; they may contain subclasses. A subclass is governed by all of the true properties of its superclasses. The class concept is familiar to many, particularly those who have studied object-oriented programming. It addresses three of the requirements outlined above.

[2] An example is an adjacency matrix as a partial representation of a layout of rectangles.

1. It facilitates the definition and specification of a wide domain of spatial sets.
2. It provides a mechanism for the automatic classification of spatial sets.
3. It establishes a clear framework for extensibility.

A wide domain of spatial sets is naturally supported by the class concept. Structuring geometry into classes allows the transparent definition of useful representations and algorithms for different objects. Each class of objects, for example polygons, may have a specialized data structure or structures and a set of algorithms for computing such properties as displays, areas, convex hulls and bounding boxes. A class may partially define some its properties. For example, the location of an object in space may be unrestricted by a class, or may be restricted to a subset of space. This ability to partially define properties supports the meaningful creation of instances of a class. An instance displays all of the properties defined for its class, but holds specific values for properties that have been only partially defined by the parent class.

The automated classification of spatial sets may be supported by algorithms which operate on members of a class and graduate those members down the subclass tree if they meet the conditions specified by a subclass. This classification can be extremely efficient, in that its complexity is proportional to only the depth of the class tree, not to the number of classes defined.

Extensibility is addressed by the class concept as new classes may be defined by specification of additional properties on an existing class or set of classes. All of the properties of the old classes are maintained in the new class.

3.2.2. Geometric Abstractions

Geometric abstractions are representations of only a portion of the properties of a spatial set. A geometric abstraction may have its own representation and set of algorithms for the computation of properties represented by that abstraction. For example, the vertex-edge graph of a polyhedron can be simply represented in a variety of ways and is sufficient for wireframe display. Geometric abstractions are organized into a hierarchy in which an abstraction is a sub-abstraction of another in the hierarchy if it can be computed from the higher level abstraction. Figure 3-1 shows an abstraction hierarchy for some simple abstractions on an assembly of geometric objects.

The concept of geometric abstractions contributes to six of the eight requirements for geometric reasoning:

1. It provides a means to represent entities within the modelling domain prior to having complete knowledge of their configuration.
2. It provides a uniform representation for relationships between sets of objects.
3. It unifies queries on objects with representations of objects.
4. It provides a framework for the formulation of generation problems.
5. It provides a framework for the maintenance of consistency of geometric information under modification.
6. It provides transparent access to appropriate levels of representation.

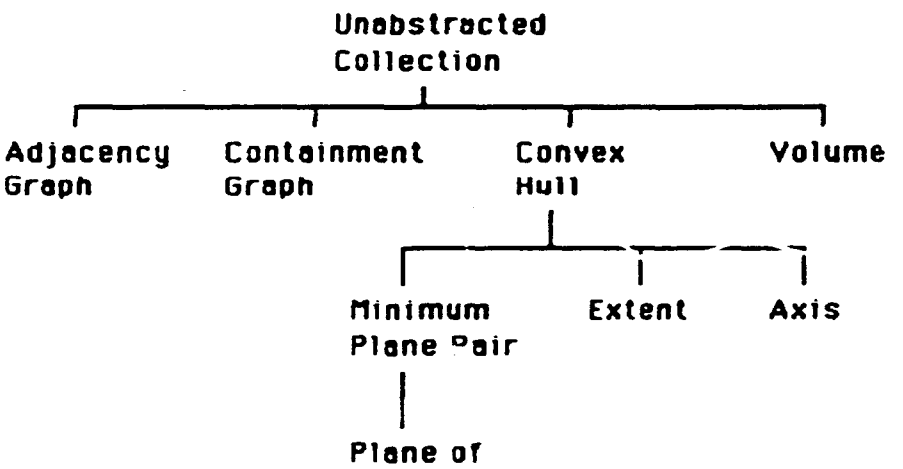

Figure 3-1: An abstraction hierachy for collections of polyhedra

The domain of representation discussed in Section 3.1 is a specification of the classes of spatial sets that should be modelled by a geometric reasoner. It does not give guidance for creating the representations which support modelling. Geometric abstractions provide a means to build multiple representations, and to link those representations together into a hierarchy. Multiple representations support the incomplete definition of geometric entities as each of the representations can be considered a partial model, which captures some properties but leaves others undefined. Linking these multiple representations together into a hierarchy provides a framework for inference upon partial knowledge. Figure 3-2 demonstrates how an abstraction hierarchy supports this inference. If all that is known

about a particular geometric entity is its convex hull, then several different abstractions (minimum plane pair, plane-of, extent and axis) may still be computed. Computation of additional information demands that information be added to the reasoner in some fashion.

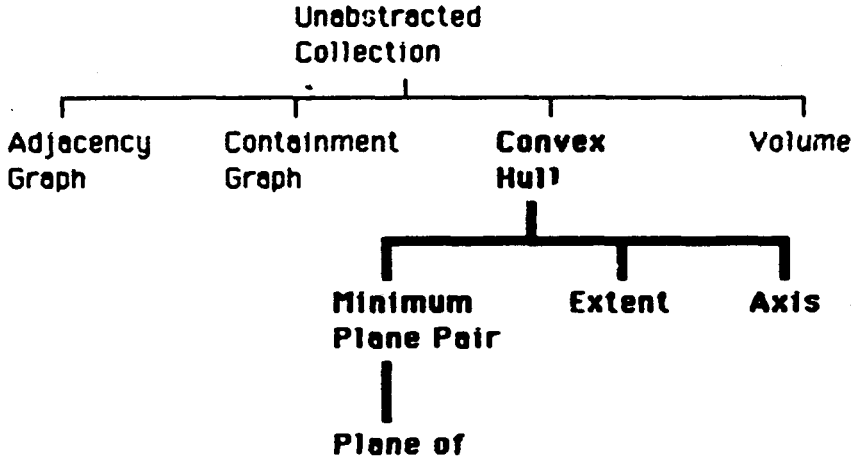

Figure 3-2: Use of an abstraction hierarchy for inference on incomplete information

Information about relationships between objects may be captured in a graph representation which is independent of the relationship modelled. The only difference between different relationships is the algorithms which are used to examine the graph data structure[3]. For example, a graph which captures information about adjacencies can be searched transitively to determine connected sub-components. A graph which captures proximity information cannot be so searched as proximity is not a transitive relation.

Geometric abstractions unify notions of queries and partial representations. A query on a geometric entity produces an abstraction or partial representation of that entity. This abstraction may be geometric object in its own right, or may be a graph representation or a vector or scalar quantity. In any case, for every query on a geometric entity there exists an entry into the abstraction hierarchy for that object.

An abstraction hierarchy can be interpreted in two directions. If *downward computation* is defined as the direction from complete models to less complete models, and *upward computation* is the reverse, then downward computations correspond to queries on models and upwards computations correspond to generation algorithms. For each query there can be formulated a problem in generation, which may or may not have a reasonable computational solution. In almost all cases the algorithm will be combinatoric in complexity. For example, generation of rectangular layouts from an orthogonal graph representation produces on the order of 10^6 possible layouts when only 10 rectangles are considered. This dual interpretation of abstraction hierarchies links together queries and generation problems.

The abstraction hierarchy provides a means for maintenance of consistency under modifications. A change to one of the representations in an abstraction hierarchy may impact information contained in other representations. A conservative strategy is to delete all information dependent upon the changed information and to recompute the deleted information as it is needed. The abstraction hierarchy can be used to incrementally improve upon that conservative strategy by including specific modification algorithms to selectively update nodes in the hierarchy. For example, under isometry transformations, the convex hull of an object will be changed by exactly the exactly the same transformation that applies to an unabstracted collection.

When a query is put to a model stored using an abstraction hierarchy, the hierarchy is searched for a representation from which that query can be answered. If the representation is not found, it is computed. If the required representation cannot be computed due to insufficient information being stored in the hierarchy, then that fact is returned as the response to the query. All of this can occur in a manner completely transparent to the application which is using the reasoner.

3.2.3. Features

Features[4] capture the nearly hierarchical decompositions of composite objects that humans impose upon these objects. This decomposition is essential to the process of understanding, manipulating and creating geometric forms, yet it has little to do with geometry *per se*. It is rather a recursive naming mechanism, that supports the aggregation of geometric entities into complex composites. This naming mechanism should be as flexible and general as possible, to support many different approaches to decomposition. Virtually the only restriction that is reasonable to place on this mechanism is that it not be self-referential, that is, that no

[3]These could be called inference algorithms without doing violence to common usage.

[4]The origin of the term "features" arises from its general usage in computer-aided design as a means of specification of portions of a model that are of interest. Recently specific use of the term has occurred in the context of mechanical engineering design and machining [Libardi 86] [Nielson 86]. Its meaning here does not contradict these usages.

feature can contain itself as one of its sub-features. The concept of features addresses four of the global system requirements:

1. It adds the notion of an assembly of objects into the domain of representation.
2. It provides a framework for the computation of relationships between objects.
3. It provides a framework for the computation of queries on objects.
4. It is a basic interface mechanism, allowing uniform access to a system.

The fundamental domain of representation for a geometric reasoning system is subsets of $\mathbf{R}^3$. Features provide a mechanism to structure combinations of these sets in any way required by an application. The single limitation on features, that of non-self-referentiality, imposes very *little* on the scheme. Features may be used to represent strictly hierarchical structures such as the instance tree of modelling systems [Woodbury 86] and high performance graphics packages [Brown 85] as well as structures which approach a general data access scheme.

By providing a means to group together geometric entities of interest, features provide an mechanism for passing arguments to functions which compute relationships between entities. For example, polyhedra which represent a robot arm may be grouped together under a single feature and submitted to an algorithm which computes the allowable movements of the assembly from the constraints imposed by the joint geometries. If each of the links of the robot arm is represented by a number of polyhedra, these may be grouped into a sub-feature and considered as one unit for purposes of the allowable movement algorithm. The information computed by the algorithm is a property of the feature as a whole, not of the individual spatial sets which compose the feature.

In a similar fashion, features may be used for the computation of queries on objects. For example, the convex hull of a feature which is composed of a number of spatial sets will in general be different from any of the convex hulls of the component sets. Queries on features compute properties which pertain to the entire assembly, not necessarily to its constituent parts.

Features are a basic access mechanism to a geometric reasoner. They provide a means to reference geometric objects in terms of the semantics of the application at hand, rather than in terms of data structures which represent geometric sets. A user need know little more than the basic structure of features and the protocol of operations which apply to them to effectively build and access complex geometric models.

3.3. The Concepts Together: An Architecture

The three concepts, classes of spatial sets, abstractions and features mutually interact. Out of these interactions can be developed an overall system architecture.

The two concepts of classes of spatial sets and features are related by a single rule: a feature may be declared as a member of a particular class of spatial set. This rule has two implications:

1. It preserves the uniformity of features as the main access scheme to the geometric reasoning system. The specific geometric models which may be part of the definition of a feature are hidden from the user of the reasoner. All that is visible is the feature hierarchy.
2. It further restricts the allowable contents of a features. A feature which is also a spatial set may contain sub-features, but these sub-features are constrained to be elements from the boundary of the spatial set[5]. For example, a window regulator[6] for an automobile may be described as having several parts, one of which is the backplate. If the backplate is declared to belong to the class of polyhedra, then all sub-features of the backplate must be a part of the boundary of the polyhedron which describes the backplate.

The interaction between features and abstractions can be expressed by a single rule: Features are the fundamental units of abstraction. This rule has two implications:

1. Only features may be abstracted. An abstraction hierarchy grows whenever queries are performed upon an entity. Since features are the only entities upon which queries are defined, they are also the only entities which support abstractions.
2. Features store abstraction information at the appropriate level. An abstraction generated on a particular feature is stored in the abstraction hierarchy associate with that feature. Any abstractions generated in the process of computation are automatically stored at the appropriate level in the feature hierarchy. For example, consider Figure 3-3. Representations which abstract the window regulator as a whole are stored with the feature labelled *window regulator*. Representations which abstract only the backplate of the window regulator are stored with the feature labelled *backplate*.

Classes of spatial sets and abstractions are simply related. The rule in this case is: every abstraction is represented by either a spatial set, a graph, a vector or a scalar. This rule has a single implication:

1. It makes all of the power of the underlying operations on spatial sets available to the abstraction mechanisms in the geometric reasoner. For example, in computing the bounding box of an assembly of objects, the reasoner could proceed

---

[5] This restriction on allowable contents creates a close approximation to most implementations of part-whole hierarchies.

[6] A window regulator is the device, inside the door of an automobile, which moves the glass of the window in response to turning the window crank or to pushing the activator switch in electrical models.)

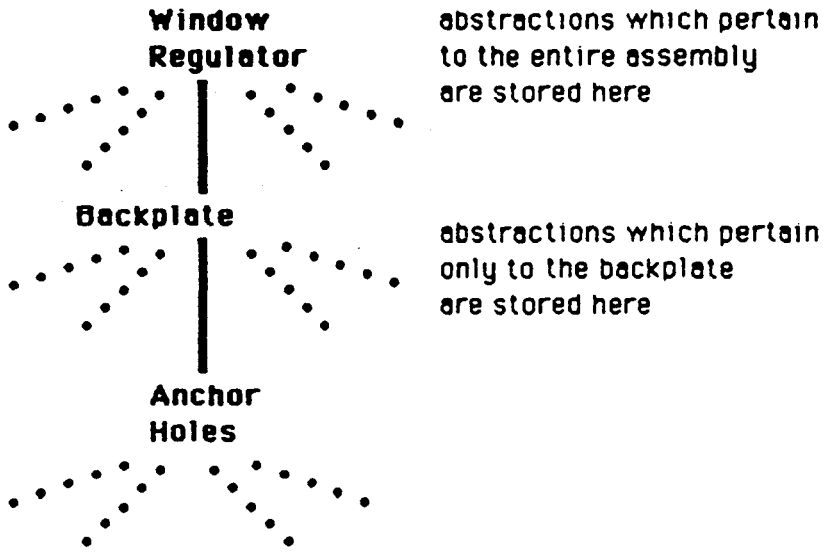

Figure 3-3: Storing abstractions at the appropriate level

from either the complete assembly or could first compute the convex hull of the assembly. In either case, by virtue of both representations being defined in terms of polyhedra, the same algorithm would apply.

When the three concepts are considered simultaneously an overall system architecture emerges. Figure 3-4 diagrams this organization In this figure, the boxes refer to each of the concepts and the lines are inheritance relations between concepts. The top level box, labelled *object* denotes the unity of all of the concepts as data objects in an implementation.

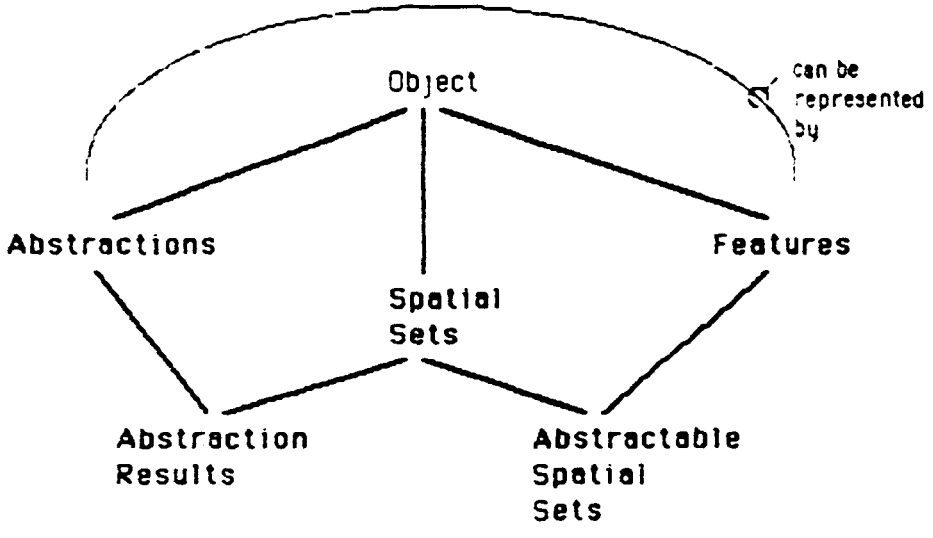

Figure 3-4: The overall system architecture

3.4. Implementation

This implementation of this architecture is affected by the choice of programming environment. We chose object oriented programming for this exploration as its concepts of polymorphism and inheritance map naturally onto the hierarchical structuring of geometric information that is proposed here. The particular programming environment used, CommonLoops, has two special properties, multiple inheritance and multi-methods, that affect the overall system architecture:

1. Multiple inheritance permits facile merging of the system concepts. The class *abstractable spatial sets* is formed by inheriting from the classes *features* and *spatial sets* and by defining a small number of methods specific to the new class.

2. Multi-methods permit the specialization of methods on the basis of class membership of more than one of the method

arguments. This is in distinction to so-called *classic message passing* in which only the class of the first argument is significant. Multi-methods simplify the creation of uniform protocols over operations which involve more than one object.

We have implemented a simple two-dimensional geometric reasoning system based upon the three concepts, classes of spatial sets, geometric abstractions and features. With the lessons learned from this exercise we are currently implementing a much larger scale demonstration in three dimensions, using an existing solids modelling system as the basis for geometric computation.

4. Conclusions and Recommendations

The *architecture for geometric reasoning* contained in section 3 is offered as a promising approach to representation and manipulation of geometric information for knowledge based systems. Its implementation is underway, and it will be evaluated for problems of robot manipulation. It will also be applied to problems of mechanical parts description for manufacturing or assembly.

Additional approaches to geometric reasoning, offered by colleagues of the authors, are partly outlined here. One approach is an investigation of natural language function in conveying geometric information; this work has its origins both in cognitive psychology and in architectural research into spatial cognition [Goel 86]. Another approach, convincingly demonstrated in two-dimensional space, is the representation and abstraction of loosely packed arrangements of rectangles in automated layout research [Flemming 86]. A third apprach is reflected in language design for geometric representation and manipulation [Wing 85]. A fourth approach is found in studies of automated assembly and disassembly of objects. A fifth approach is exemplified by the design of domain models for robot manipulation within complex facilities such as nuclear power plants, together with the design of blackboards for robot control in comparably rich environments [Keirouz 86]

5. Acknowledgements

The main research approach presented herein (section 3) is supported by the Electric Power Research Institute under contract RP2515-1, "Robotics in Construction and Maintenance." The authors are pleased to acknowledge the support, interest, and guidance of EPRI and Dr. Floyd Gelhaus. We are especially grateful for permission to discuss this approach with the research community prior to its conclusion and delivery to EPRI later this year, and point out that the results have not yet been reviewed or checked for accuracy.

References

[Akiner 85] V.T. Akiner. Topology - 1: A Reasoning System for Objects and Space Modelling Via Knowledge Engineering. In *Design Engineering Division Comference on Mechanical Vibration and Noise*. American Society of Mechanical Engineers, Cincinnati, Ohio, September, 1985.

[Borning 79] Alan Borning. *ThingLab- A Constraint Oriented Simulation Laboratory*. Technical Report, Palo Alto Research Center, July, 1979.

[Brooks 81] Rodney Allen Brooks. *Symbolic Reasoning Among 3-D Models and 2-D Images*. PhD thesis, Stanford University, June, 1981.

[Brown 85] Maxine Brown. *Understanding PHIGS*. Template, The Software Division of Megatek Corp., 9645 Scranton Rd., San Diego, Ca., 92121, 1985.

[Davis 81] Ernest Davis. *Organizing Spatial Knowledge*. Technical Report 193, Yale University, Computer Science Department, January, 1981.

[Davis 86] Ernest Davis. *Research Notes in Artificial Intelligence: Representing and Acquiring Geographic Knowledge*. Pitman Publishing Ltd., 128 Long Acre, London, England, WC2E 9AN, 1986.

[Eastman 79] Charles M. Eastman and Kevin Weiler. *Geometric Modeling Using the Euler Operators*. Technical Report 78, Institute of Physical Planning, Carnegie-Mellon Univ., February, 1979.

[Fahlman 74] Scott Elliot Fahlman. A Planning System for Robot Construction Tasks. *Artificial Intelligence* 5(1):1-49, 1974.

[Fikes 71] R.E. Fikes and N.J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2:189-208, 1971.

[Flemming 86] Ulrich Flemming, Michael D. Rychener, Robert F Coyne, Timothy J. Glavin. *A Generative Expert System for the Design of Building Layouts: Version 1*. Technical Report, Center for Arts and Technology, Carnegie-Mellon University, June, 1986.

[Goel 86] Vinod Goel. Assigning Locative Prepositions to the Spatial Relations Implicit in 3-D Geometrical Models of Objects and Scenes. Master's thesis, Faculty of Environmental Studies, York University, Toronto, Ontario, Canada, July, 1986.

[Gosling 83] James Gosling. *Algebraic Constraints*. PhD thesis, Computer Science Department, Carnegie-Mellon University, May, 1983.

[Keirouz 86] Walid T. Keirouz, Daniel R. Rehak, Irving J. Oppenheim. Object-Oriented Domain Modeling of Constructed Facilities for Robotic Applications. In *Applications of Artificial Intelligence in Engineering Problems, 1st International Conference*, pages 141-150. Springer-Verlag, Southampton University, U.K., April, 1986.

[Klahr 82] Philip Klahr, David McArthur and Sanjai Narain. Swirl: An Object-Oriented Air Battle Simulator. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 331-334. American Association for Artificial Intelligence, 1982.

[Kuipers 77] Benjamin Jack Kuipers. *Representing Knowledge of Large-Scale Space*. PhD thesis, Department of Mathematics, Massachusetts Institute of Technology, July, 1977.

[Libardi 86] Eugene C. Libardi, John R. Dixon, Melvin K. Simmons. Designing with Features: Design and Analysis of Extrusions as an Example. In *Spring National Design Engineering Conference*. American Society of Mechanical Engineers, Chicago, Illinois, March 24-27, 1986.

[Light 82] R. Light, D. Gossard. Modification of geometric models through variational geometry. *CAD - Computer Aided Design* 14(4):209-214, July, 1982.

[Lynch 60] Kevin Lynch. *The Image of the City*. MIT Press, Cambridge, Mass., 1960.

[Nielson 86] Eric H. Neilson, John R. Dixon, Melvin K. Simmons. *How Shall We Represent the Geometry of Designed Objects?*. Technical Report, Department of Mechanical Engineering, University of Massachusetts, 1986.

[Preparata 85] Franco P. Preparata and Michael Ian Shamos. *Texts and Monographs in Computer Science: Computational Geometry - An Introduction*. Springer-Verlag, New York, N.Y., 1985.

[Requicha 80] Aristides A.G. Requicha. Representation for Rigid Solids: Theory, Methods and Systems. *Computing Surveys* 12(4):437-464, December, 1980.

[Simmons 82] Reid F. Simmons. Spatial and Temporal Reasoning in Geologic Map Interpretation. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 152-154. American Association for Artificial Intelligence, Carnegie-Mellon University, Pittsburgh, Pa., August, 1982.

[Steele 80] G.L. Steele Jr. *The Definition and Implementation of a Computer Programming Language Based on Constraints*. PhD thesis, Massachusetts Institute of Technology, August, 1980.

[Sussman 80] G. Sussman and G. Steele. CONSTRAINTS - a language for expressing almost hierarchical descriptions. *Artificial Intelligence* 14(1):1-40, August, 1980.

[Sutherland 63] I.E. Sutherland. *Sketchpad: A Man-Machine Graphical Communication System.* Technical Report 296, MIT Lincoln Lab., 1963.

[Wing 85] Jeannette M. Wing, Farhad Arbab. *Geometric Reasoning: A New Paradigm for Processing Geometric Information.* Technical Report cmu-cs-85-144, Computer Science Department, Carnegie-Mellon University, 1985.

[Woodbury 86] Robert Woodbury. VEGA: A Geometric Modelling System. In *IBM Advanced Educational Projects Conference.* IBM, San Diego, Ca., April, 1986.