Robert F. WOODBURY
Department of Architecture,
Carnegie-Mellon University
Walid T. KEIROUZ,
Irving J. OPPENHEIM,
Daniel R. REHAK
Department of Civil Engineering,
Carnegie-Mellon University

Geometry and Domain Modelling for Construction Robots

Abstract: Construction robots operate in an environment very different from that of manufacturing robots. This environment is less structured, more complex and more dynamic than is the norm in manufacturing. In addition, construction robots are inherently mobile, as they are engaged in building or maintaining an immobile structure which is large compared to their dimensions. Another complicating factor is the uniqueness of actions that must be taken by a construction robot: the number of special conditions that may exist in buildings is large. All of these differences provide arguments for two related capabilities that are required of construction robots: the ability to *reason about* and to *model* their environment. In this paper we present current work at Carnegie-Mellon University which addresses the problems of *geometric reasoning* and *domain modelling* in the specific context of *knowledge based expert systems.*

1. Introduction

Construction robots operate in an environment very different from that of manufacturing robots. This environment is less structured, more complex, more dynamic[1] than is the norm in manufacturing. In addition, construction robots are inherently mobile, as they are engaged in building or maintaining an immobile structure which is large compared to their dimensions, as distinct from common factory operations where the smaller workpiece may be brought to the robot. Another complicating factor is the uniqueness of actions that must be taken by a construction robot: the number of special conditions that may exist in buildings is large. All of these differences provide arguments for two related capabilities that should be developed for construction robots, these being the ability to *reason about* and to *model* their environment.

The environment of construction robots is a physical one; it consists of physical objects related to each other in a wide variety of ways for a wide variety of purposes. Reasoning about this environment almost always involves using information about the geometry of the environment. Existing *knowledge based expert systems*, which can reason after a fashion, do not provide adequate means to include geometric information in their reasoning processes.

The use of geometric information in a *knowledge based* context is presumed to find future use in both construction robotics and in architectural design. It then poses several requirements for the representation and manipulation of geometry:

Abstractions of Geometry
: The complete geometry of an object often contains too much information to use effectively in a reasoning process. Suitable abstractions which capture the key information and suppress all else are required.

Inheritance of Geometric Properties
: One of the principle ideas behind knowledge representation is that of inheritance. The full implications of inheritance in a geometric domain remain to be discovered, but certain important geometric properties are clearly recognized as inheritable.

Geometric Production Rules
: Another principle idea in knowledge based systems is that of productions, or rules. Current production systems operate at too low a level of abstraction to be convenient for design or construction applications. A *geometry based* production system would provide the needed ability to express productions pertinent to the physical world of design and construction.

Geometric Constraints
: An operation made on one part of a geometry model may change other parts of the model in unanticipated ways. Mechanisms for the automatic enforcement of constraints on geometry are required.

Automatic Generation of Geometry
: Geometric information is independent of the domain in which it is applied. Thus a single representation can be used across many applications. Provably exhaustive automatic generation facilities for geometry are complex and time-consuming to build. A core of generation programs which could aid in a wide variety of applications is required.

A *model* of an environment can provide powerful assistance in performing tasks within that environment. Such a *domain model* organizes information about a robot's world so that the robot may intelligently and quickly

[1] Dynamic has two meanings in the construction environment. The environment is both physically dynamic in the sense that the physical constituents of a site change and conceptually dynamic (or evolving) in that the information known about a project changes in both kind and amount.

access needed information while ignoring irrelevant information. Domain modelling also provides a means to test operations and to optimize those operations in computer memory, prior to committing them to physical action by a robot. An effective domain modelling system for construction robotics has several key requirements:

Representational Abilities	A domain model must contain a description of the facility in terms of its geometry, its non-geometric characteristics and in terms of the operations that can be performed upon various parts of the facility.
Efficiency	A domain model must be able to deal with very large amounts of information with great efficiency.
Model Validity	A domain model must have have an ability to update its contents as the world it is modelling changes.
Consistency	A domain model must maintain its own internal consistency.
Data Abstraction	A domain model must provide abstract views of the information it contains.

In this paper we discuss current work at Carnegie-Mellon University that is aimed at understanding these issues and at creating systems to provide effective software for construction robotic applications. In particular we discuss the following issues:

- Geometric information in knowledge based systems
  - The architecture of geometric modelling in knowledge based systems.
  - Abstractions of geometry as the basis for geometric reasoning.
- Domain modelling for construction robotics
  - Network based representations for domain modelling.
  - Object-oriented programming languages for domain modelling.

2. Geometry Modelling

2.1. Architecture of Systems

The abstraction of *geometric information* from the total information describing the world is a natural human process in the fields of design and construction. To develop powerful *knowledge based systems* geometric information must be available as a basic data type for representation and reasoning. However, current expert system building tools are inadequate. At best, these systems provide certain data structuring facilities adequate to create geometric representations; at worst they provide only representations of atomic symbols. The developers of expert systems currently must themselves resolve the issues of the geometry to be represented and the representation to be structured. Thus system developers are forced to do much of their work at a level of abstraction well below that of the problem they are trying to solve; they are put in the position of having to create basic tools before they can proceed.

The lack of good geometric modelling and reasoning facilities in knowledge based systems has been strongly felt in numerous and broad ranging expert system development efforts. Fahlman, in BUILD [Fahlman 74], by his own attestation, spent about 80% of his time developing a modelling scheme for polyhedral objects. Baker [Baker 85] also spent an inordinate amount of his effort dealing exclusively with geometric information. It seems obvious that powerful facilities for the representation and manipulation of geometry are required for knowledge based systems for use in design and construction. Less obvious is the "what" and "how" of these geometric computation facilities. There are several aspects to this problem:

- The architecture of the representation.
- The interface between geometry and knowledge based systems.
- The information represented.

In this section we discuss a particular approach to these problems, beginning with these observations:

- Many geometric operations are extremely well defined. The volume and mass properties of objects have precise mathematical definitions as do many relationships between objects, such as adjacency and separability. There exist efficient (and often optimal) algorithms for the computation of many of these properties.
- Many geometric operations and questions are used very widely in the course of solving a problem. For example, two objects may be combined many times, or a kinematic mechanism may be moved through a large number of motions in the search for a problem solution.

These observations provide arguments for the efficient implementation of new underlying programs to perform the geometric modelling function. One persistent difficulty with expert systems is their speed, particularly when they perform search. This speed problem is likely to persist because the types of problems for which expert systems are being written are difficult and are perhaps intractable. If expert systems are to use geometry operations as one of their fundamental data types, then those operations should be implemented as efficiently as is possible. Today, efficient operations are implemented as algorithms. Thus the basic *computational facilities* for geometric modelling should be implemented in algorithmic languages.

The way in which geometric modelling tools are presented to system builders will greatly affect their usefulness. Knowledge based systems have utilized concepts which are now familiar to developers of applications. Key amongst these concepts are those of *search, inheritance* and *production rules.* If geometric modelling tools were presented to system developers using this same set of concepts then the process of using geometric information would become much simpler. There exist ideas, some developed to be paradigms, some very nascent, for the expression of geometric information in each of these conceptual areas. In *search,* the idea of programs which can exhaustively generate configurations of equivalence classes of geometry has been used in the development of several programs and for several studies of form [Flemming 85] [Galle 81]. Some preliminary work has been done in the area of *inheritance* in the representation of classes of objects defined by assemblies of generalized cylinders, [Brooks 81]. Finally the shape grammar paradigm demonstrates an approach to the development of *production systems* which operate on geometric information [Stiny 80].

2.2. Abstractions of Geometry

For geometric information, itself an abstraction of the world, the definitive representation is of regions of space in three dimensional Euclidean space. If such a representation is unambiguous and has a means of validity testing, then it constitutes a complete representation of geometry, from which all geometric information may be extracted. Such a complete representation must satisfy two criteria if it is to be useful in this field.

- It must allow for flexibility of definition.
- It must be an efficient and convenient medium upon which to perform computations.

2.2.1. Flexibility of Definition

Many modelling processes in the design and construction of buildings develop descriptions incrementally as they proceed. Top down design is a cogent example, in which an object is specified at first incompletely and later at increasing levels of detail. These levels of detail involve information of different kinds as well as in widely different amounts. For example, in the early stages of building design, all that may be known are adjacencies and rough sizes. Later in the process of design, sizes and boundaries may be more precise and the dimensions of

objects such as walls, furnishings and mechanical systems may be known. Analogous situations occur in assembly planning, cost estimating and task planning. A modelling system which can faithfully represent information at various stages in the design and construction process must be able to form valid models of the world regardless of the information (or lack of it) that is available.

In constrast, current systems for modelling geometry demand precise knowledge of a form before a model can be created. For example, in solid modelling systems, one must generally know the exact dimensions of an object prior to its input to the system. However, there do exist *formalisms* for modelling situations in which only partial information about an artifact is known. These formalisms are *abstractions* of geometric information. The task then becomes one of discovering a set of geometric formalisms and linking them together in an appropriate way so that they provide a useful medium for the representation of a developing artifact or world view. Links between levels in such a modelling system can be viewed as algorithms or manipulations of the model which operate within and between representations. These algorithms can be thought of as being in the classes of *generation, modification, queries* and *classification*.

Multiple levels of abstraction are required if geometric knowledge is to be developed incrementally, in a top down manner as knowledge about an environment grows. At higher levels of abstraction, available geometric information may be limited to relational information between objects of unknown shape and size. As the amount and kind of known information expands, it becomes less abstract and requires different types of more specific abstractions.

2.2.2. Efficiency of computation

When a computation is performed upon a model, the speed with which that computation executes is often dependent upon the organizational structure of the model. An example comes from boundary modelling of solid objects. A complete boundary model of a solid object need only contain information which relates faces, edges and vertices in a methematically coherent way. However, most boundary modellers include additional information designed to make certain, frequently performed computations more efficient. An example is the list of edges maintained by boundary modellers which makes interactive display of a model much faster. The additional structure imposed by this added information is an *abstraction* from the total of the information available. Many of these abstractions, useful for particular computations and in general invisible to the user of a modelling system, are required to make a modeller perform in an efficient manner.

An example of this type of abstraction is one which supports queries on the spatial location of objects. One possible organization of geometric objects is a list of all such objects with the location of the object stored with the object in the list. In this scheme a query which determines if two objects overlap will require $O(n^2)$ comparisons (each of which may be complex) as each object must be compared against every other object in the system. Organizing the location information of objects such each object is stored such that it can directly access objects close to it ean change this $O(n^2)$ cost to an $O(n)$ cost or even to an $O(1)$ cost at the expense of performing checks at model creation time. This type of spatial localization can greatly increase the time order efficiency of many queries. Many authors have reported on research aimed at creating just such spatial localization schemes.

3. Domain Modelling

A domain model must be able to represent a given domain at two levels:

1. The *microscopic* level represents the modeling of basic objects in the domain.
2. The *macroscopic* level imposes an organization or hierarchy on the information describing the domain.

Objects and object-oriented modeling satisfy the first level of representation. All the information about an entity (both descriptive and functional) is integrated into the object representing the given physical entity. However, a structuring mechanism for organizing all the objects in the domain is still needed to render tractable the considerable details describing the modeled domain (i.e., to limit the amount of data a robot need view at any

time.). A *network-based multilayered representation* provides this mechanism.

3.1. Networks

A *network based* representation for domain modelling may be structured around the following ideas:

- The entire facility is modeled as a single model entity. The model entities can then be recursively subdivided into other entities organized into a network where nodes represent entities (individual physical objects or parts of the domain) and branches stand for connections between entities.
- The fundamental component of the model is an object which describes an entity and specifies how it can be manipulated. Four types of objects are used in the model:
  1. A *primitive* object has no subdivision. It stands for a basic physical entity which should not be subdivided. Primitive objects are defined by the modeling system and include objects such as pipes and pipe hangers.
  2. A *domain* object is recursively built from other objects and is represented as a network.
  3. A *connection* object is a branch in the network. It stands for functional links or valid paths between objects.
  4. A *virtual-space* object represents unoccupied space. Networks of virtual-space objects provide a mechanism for explicitly specifying possible paths.
- The description of a domain object may contain:
  - a network representing the composition of the object into subobjects,
  - a network of virtual-space objects,
  - a set of operators (functional information) used to manipulate the object, and
  - geometric and non-geometric descriptions (descriptive information) of the object.

The recursive subdivision of the domain model provides data abstraction and data hiding. This scheme limits the amount of information available at a given subdivision level while allowing for the access of additional detail by descending the hierarchy of networks. The recursive subdivision also provides a *focus-of-attention* mechanism. *Focus-of-attention* at a specific area can be achieved by considering only the lowest level spatial component containing that area.

3.2. Object-Oriented Modeling

Object-oriented modeling use object-oriented programming techniques to implement models. In the object-oriented model of computation, the universe is exclusively populated by entities which, although not identical, have the same basic nature. *Objects* are the exclusive inhabitants of this universe and exhibit the same basic behavior. These objects are distinct entities that communicate with each other by *sending messages*. Examples of OOPLs are SMALLTALK-80 [Goldberg 83], LOOPS [Bobrow 83], and COMMONLOOPS [Bobrow 85]. Concepts underlying object-oriented programming languages that are relevant to object-oriented models are described below.

An *object* is a complete, independent, and self-contained computational entity. It has its own private memory organized as a set of *variables* and defines a set of operations known as *methods*. As such, an object integrates data and procedures: data is stored in the object's private memory while its methods specify legal operations on the data. An object also presents itself to the outside world as a closed entity whose internals can not be accessed or manipulated by other objects. Furthermore, an object is an active entity that can compute: it invokes its methods to perform computations (e.g., to respond to messages or to draw a line on the screen). From a modeling point of view, the notion of an object as a computational entity is very rich. An object can be thought of as a physical

entity, an active process, or even a computer by itself. As such, physical entities in a domain to be modeled can be readily mapped onto objects. Objects also provide a natural mechanism for integrating descriptive and functional information: descriptive information, both geometric and non-geometric , can be stored in an object's variables; functional information, which specifies how an entity functions and what it can be used for, can be represented by an object's methods. Finally, an object can have knowledge of itself because of the self-descriptive nature of objects. Thus, the model is self-contained.

Since objects can not modify each other's internal structure, processing (change in the state of the universe) in the object-oriented universe occurs only as a result of communication between objects. *Messages* being the only form of communication represent the interactions between these objects. Thus, an object sends a message to another object requesting it to take some action. The message sent describes this request without specifying how it should be serviced. The receiver of the message can either service the request by taking some action or it can simply ignore the message. The messages an object responds to constitute the object's interface with the rest of the world. An object responds to a message by invoking one of its methods. Because of this level of indirection between messages and methods, objects are easy to modify and expand: new methods can be easily added and old ones modified without affecting other objects. Furthermore, since objects define their own methods, objects modeling a set of similar physical entities can be made to respond to the same messages while simultaneously tailoring the response to these messages to account for the particularities of each physical entity.

Each object is an *instance* of one *class*. As such, a class is a template describing the implementation of its instances. The class describes the structure of the private memory of its instances by providing a list of names for the instance variables, and specifies the messages the instances will respond to by defining a set of methods. Finally, classes are organized into an *inheritance* hierarchy. Inheritance defines the way in which properties of a class (variable names and methods) can be passed to other classes known as its subclasses. As such, inheritance provides a mechanism for specializing classes (e.g., class *Integer* specializes class *Number*; *Number* specializes class *Object*). A specialized class uses the methods of a general class through inheritance, redefinition of some methods, and addition of new methods. Classes are themselves objects and respond to messages. Thus, a class can describe the capabilities of its instances. This feature provides assistance in reasoning about objects in the model. Classes and inheritance makes it easy to add new objects to a model. Adding an object is tantamount to defining a new class which may, through inheritance, be a specialization of an already existing class.

4. Summary

We are proceeding with research in both of the areas discussed in this paper, *geometric representation and reasoning* and *domain modelling*. In the area of geometry our primary concerns are those of representation of and reasoning with geometric information at multiple levels of abstraction in an evolving domain. In the area of domain modelling we are concerned with network models for providing focus of attention mechanisms for robots and with the use of object oriented programming languages as a representation mechanism for domain objects.

References

[Baker 85] Baker, Nelson C.; Fenves, Steven J. *Simulation of the Monitor/Controller for a Robotic Excavator.* Technical Report R-85-149, Department of Civil Engineering, Carnegie-Mellon University, August, 1985.

[Bobrow 83] Bobrow, D. and Stefik, M. *The Loops Manual.* Technical Report, Xerox Corporation, 1983.

[Bobrow 85] Daniel G. Bobrow, et al. *CommonLoops: Merging Common Lisp and Object-Oriented Programming.* Technical Report ISL-85-8, Xerox Palo Alto Research Center (PARC), Palo Alto, CA, August, 1985.

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

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

[Flemming 85] Flemming, Ulrich. On the representation and generation of loosely-packed arrangements of rectangles. *Planning and Design* , December, 1985.

[Galle 81] Galle, Per. An algorithm for exhaustive generation of building floor plans. *Communications of the ACM* 24:813-825, 1981.

[Goldberg 83] Adele Goldberg and David Robson. *Smalltalk-80: The Language and its Implementation.* Addison-Wesley, 1983.

[Stiny 80] Stiny, George. Introduction to shape and shape grammars. *Environment and Planning B* 7(3):343-352, 1980.

Robert F. Woodbury
Department of Architecture, Carnegie-Mellon University,
Pittsburgh, PA 15213, U.S.A.
(412) 268-8853

Walid T. Keirouz, Irving J.Oppenheim, Daniel R. Rehak
Department of Civil Engineering, Carnegie-Mellon University
Pittsburgh, PA 15213, U.S.A.
(412) 268-2950