Interactive Generative Geometric Modeling by Geometric to Declarative Representation Conversion

Dennis Sellinger and Dimitri Plemenos
Laboratoire Méthodes et Structures Informatique
Université de Limoges
Limoges, France

Abstract

Generative modeling environments typically use a semantic description of a scene, which can be processed by a generation system to produce one or more geometric instances. In order to support an interactive environment for generative geometric modeling it is necessary to be able to convert a geometric instance into a semantic description. This paper describes such a conversion system, as part of a project to integrate the MULTIFORMES declarative geometric modeling system with a traditional interactive modeler.

The conversion algorithm attempts to reverse the hierarchical scene decomposition, used by MULTIFORMES, using a bottom-up process of sub-scene agglomeration. The algorithm has three steps; first, primitive sub-scenes are recognized and discretized, next, connected sub-scenes are agglomerated, and finally disconnected sub-scenes are associated to produce a single hierarchical representation of the scene. If knowledge about the logical organization of the scene exists, this knowledge can be applied to achieve a better semantic scene description. This knowledge may come either from a previous semantic description, its related geometric representation or directly from the designer.

Keywords: computer graphics, geometric modeling, generative modeling, artificial intelligence, declarative modeling, representation conversion, visual programming.

1 Introduction

While much effort has been invested in developing tools to support computer based geometric modeling, relatively little research has been targeted at finding ways of refocusing designers on the more creative aspects of design, by freeing them from the more tedious aspects of geometric modeling. Generative geometric modeling (GGM) is one broad area which attempts to address this problem by allowing the designer to describe a scene in a formal way and then to have an automated system generate the geometric representation(s) based on the description. Generative geometric modeling systems have been developed using a number of different description paradigms, including procedural and functional programming, ([8] and [4] respectively) and formal languages ([2]). The generative modeling system used in this paper is the MULTIFORMES declarative geometric modeling (DGM) system ([5]), which uses a rule-based description paradigm. Regardless of the description paradigm used, all GGM systems allow the designer to describe

the geometry of an object using a semantic form which embodies both geometric and non-geometric knowledge about the target design.

While GGM systems can free the designer from the more tedious aspects of geometric modeling, creating the semantic description of a scene can be difficult, especially since designers are often more skilled at sketching geometric objects and scenes than at describing them using special purpose geometric programming languages. One way to address this problem is to allow the user to describe the scene interactively, using a traditional geometric modeler, and then use an automated system to convert the sketched scene into a semantic representation. The semantic representation thus generated can then be used to generate detailed geometric representations.

This work describes a technique to convert a general geometric representation into a rule-based semantic representation which is understandable to the MULTIFORMES DGM system. Such a conversion technique forms the basis for a scene description paradigm based on visual programming by example. Using this type of visual programming, the designer can directly specify basic scene properties in an environment which is well-known and comfortable, the interactive geometric modeler, without ever having to formalize the scene description, as is required by GGM. The paper is organized in the following way. The next section describes the MULTIFORMES DGM system, with special attention paid how it is being integrated with a TGM system. Section 3 then describes two algorithms which can be used to convert a geometric instance of a scene into a declarative representation with no knowledge of the logical organization of the design other than its geometry. Once these algorithms are established several enhancements are introduced for the case where some knowledge about the organization of a scene exists. Finally, section 4 provides some concluding remarks and further perspectives on this work.

2 The MultiFormes DGM System

One of the problems with traditional geometric modeling is that the designer must often create detailed geometric representations when only a vague notion of the final design exists. In this case, the designer will typically create several preliminary designs and refine them in order to achieve an understanding of the set of possible designs. The problem with this is that many hours of tedious geometric modeling may be required to instantiate design features which may not be part of the final design. Also, the designer must rely solely on his or her experience and design skills when determining which scenes should be explored and which ones should be discarded without ever being realized.

Declarative modeling attempts to address this problem by allowing the designer to specify a scene in an imprecise and abstract way. Because the description is imprecise the design is not limited to a single geometric instance as with a traditional geometric modeling (TGM) system. Because the DGM system generates a set of valid geometric representations, less human time is spent on geometric modeling. Since a more complete set of valid geometric instances are generated, more information is available on the nature of the valid design space, allowing the designer to make better informed design decisions.

This work is being conducted as part of the XMULTIFORMES project, whose main focus is the integration of the MULTIFORMES DGM system with a tailored TGM system ([7, 6]). XMULTIFORMES is designed using the cooperative computer-aided design (CCAD) framework for integrated generative modeling systems ([3]). This paradigm is based on the principle that an automated modeling system can not generate perfect designs, but rather, superior designs are created by allowing a human designer to guide the

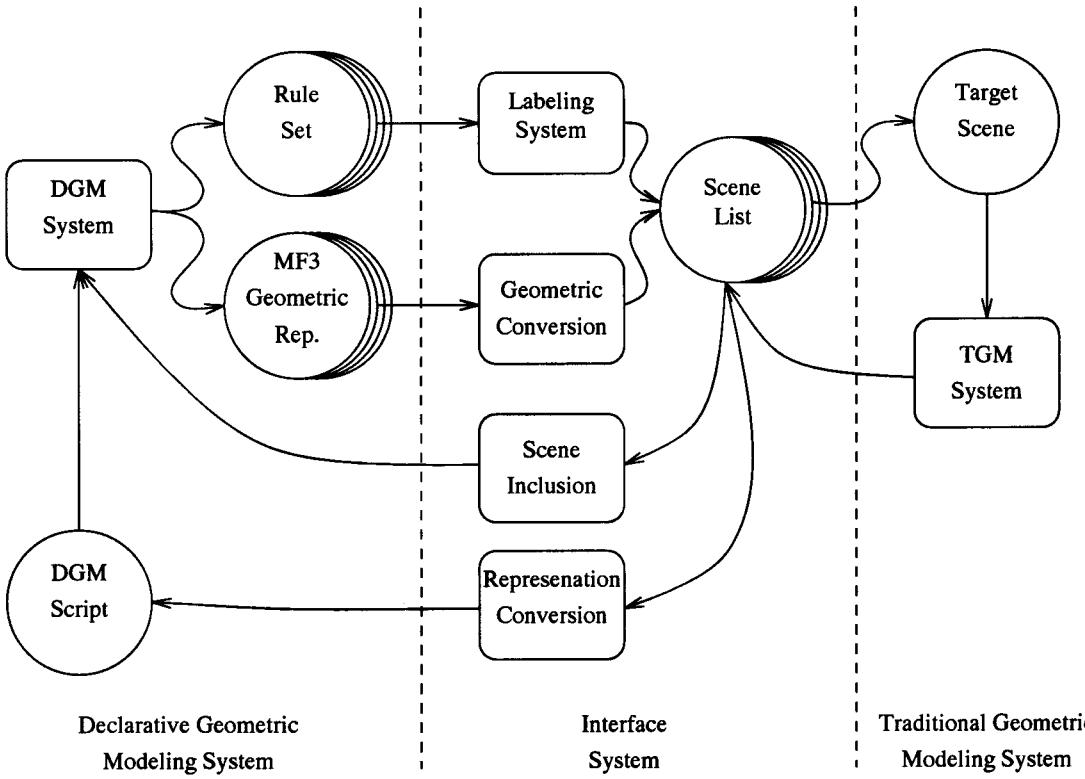


Figure 1: Data flow representation in XMULTIFORMES geometric modeling sub-systems.

generative system through successive rounds of automated modeling. To support this iterative design path, geometric and non-geometric information must be allowed to flow between the two different modeling systems. For example, figure 1 shows the flow of information between the XMULTIFORMES geometric modeling sub-systems.

XMULTIFORMES includes three main geometric modeling sub-systems, the DGM and TGM sub-systems, and an interface sub-system to allows them to communicate. The DGM system takes a rule-based scene description as input and generates a set of scenes which it maintains as a set of rules and a basic geometric representation. The interface system uses this information to generate a set of geometric representations which are more suited to interactive modeling and uses the non-geometric information, available from the set of rules, to assign labels to sub-scenes. The XMULTIFORMES TGM system is specifically designed for manipulating scenes generated by the DGM system. The TGM system reads scenes from a list which is maintained by the interface system. After interactive modeling, the updated scene can be returned to the scene list.

In order to support the iterative nature of CCAD modeling XMULTIFORMES allows the designer to create and modify designs from either the DGM system, using a MULTIFORMES script, or the TGM system. In order to perform automated generation using scenes from the TGM system the designer may use either the scene inclusion process to insert previously instantiated geometric representations, verbatim, into scenes created in the next round of automated generation or the representation conversion process which converts a geometric instance into a MULTIFORMES declarative description. In this case, the designer interactively sketches an example the desired design with the TGM system, which is then automatically converted into a declarative representation and processed by the generation system. It is in this sense that the representation conversion process facilitates interactive generative modeling, since it allows the user to intervene in the

generation process and change the scene description by interactively manipulating its geometric representation.

One of the problems of representation conversion is that since the geometric coverage of the TGM system is larger than that of the DGM system, conversion is necessarily an approximation. Also, since the sets of geometric instances created using two different declarative representations may overlap, the generated declarative representation is not unique. The representation which is actually produced depends on the conversion algorithm and on whether some knowledge about the logical organization of the scene exists independent of the geometric representation.

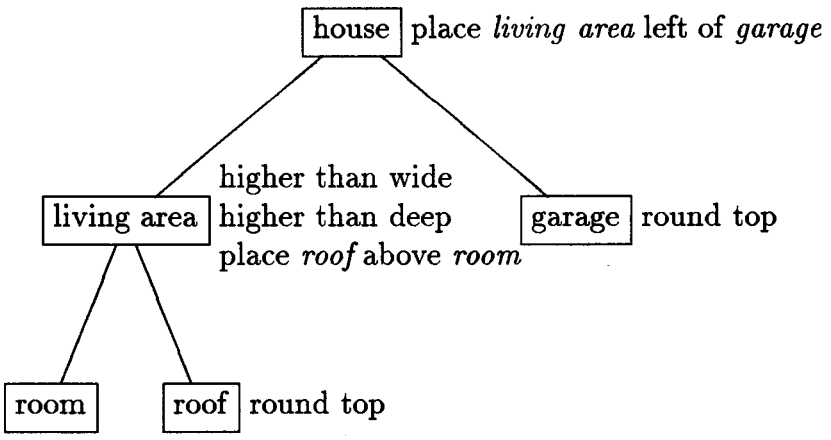
2.1 Hierarchical Scene Description

One of the problems of declarative modeling is that complex scenes are often difficult to describe. To address this problem MULTIFORMES uses a hierarchical scene decomposition technique. Using this method, rather than describing the entire complex scene as a single unit, it is decomposed into a series of sub-scenes, which are more easily described. The technique also requires that the relationships between the various sub-scenes be described. If a sub-scene is still too complex it can be further decomposed into sub-sub-scenes and the process repeated. The result is a hierarchical scene description represented by a tree whose nodes are sub-scenes and whose edges are the relationships between sub-scenes.

MULTIFORMES implements two basic types of rules, inter-dimensional and inter-scene. A sub-scene can be described using local inter-dimensional rules of size and deformation, while the relationship between two sub-scenes is specified using global rules of inter-scene size and position. For example, figure 2 (a) shows an abstract, logical representation of a scene. This description decomposes a *house* into its various sub-parts and then uses rules to describe the parts and the relationships between them. This type of description is called the external declarative representation (EDR). The EDR can be translated, unambiguously, into a rule-based representation (see figure 2 (b)) called the internal declarative representation (IDR). MULTIFORMES parses the IDR to construct a hierarchical representation of the scene as a set of constraints on the bounding boxes of each sub-scene. These constraints are then used to generate a set of geometric instances of the scene as shown in figure 2 (c).

MULTIFORMES generates a set of valid geometric instances from a set of rules and a fact-to-establish (FTE). In the example given in figure 2, the FTE is initially *house*. In the first part of the generation process, MULTIFORMES uses an inference engine to generate a set of local constraints for the bounding box of each sub-scene from the inter-dimensional rules, and augments the set of global constraints using the inter-scene rules. Next, MULTIFORMES enumerates all possible scenes and tests them against the set of generated constraints. When a valid scene is found, those rules which do not affect the size and position of the bounding box of the sub-scene (typically rules of deformation) are applied.

The geometric representation generated by MULTIFORMES is not well suited for interactive geometric modeling, thus, after a valid scene has been generated its geometric representation is converted to a more appropriate form. This representation consists of four geometric levels implemented in three linked-lists ([7]). At the lowest level is a list of vertices which are used to implement a set of geometric primitives, the second level of representation. At the third level is the MULTIFORMES compound primitive, which is a deformable parallelepiped constructed by arranging six Bezier patches into a cube.



(a)

House(x,p) → CreateFact(x,p)
LivingArea(y,x)
Garage(z,x)
PositionLeft(y,z,x);

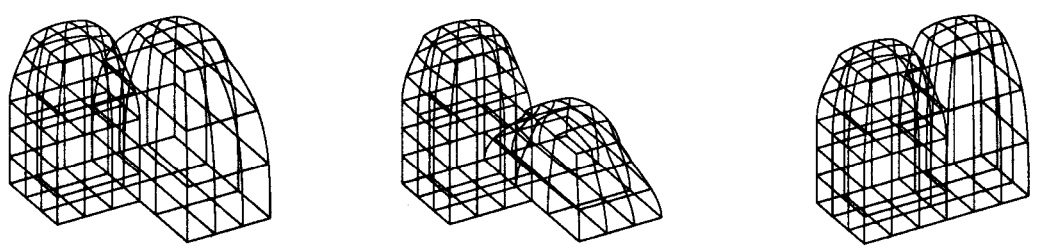
LivingArea(x,p) → CreateFact(x,p)
HigherThanWide(x)
HigherThanDeep(x)
Room(y,x)
Roof(z,x)
PositionAbove(z,y,x);

Garage(x,p) → CreateFact(x,p)
RoundTop(x,80);

Room(x,p) → CreateFact(x,p);

Roof(x,p) → CreateFact(x,p)
RoundTop(x,80);

(b)



(c)

Figure 2: Design representations used by MULTIFORMES; the external declarative representation (a), the internal declarative representation (b) and several geometric instances (c).

This compound primitive is the basic building block used by MULTIFORMES to construct scenes. At the highest level is the object entity, which is constructed from either a set of primitives or other objects. The compound primitive is a special case object which is only important because of its relationship with the generation system. Objects can be used to preserve the hierarchical nature of the scene by allowing sub-scenes to construct sub-objects. Since MULTIFORMES can only be used to construct scenes which can be represented by an acyclic graph (i.e. a tree), no cycles can be generated in the object definitions and thus, MULTIFORMES can only generate valid geometric object representations.

The IDR implies much information which is not strictly geometric. Since it is important that this non-geometric information be retained, a system of geometric entity labeling has been developed ([7]). The labeling system attempts to assign a package of information to each geometric entity at every geometric level based on its creation rules and on a set of user specified labels. This information may prove to be very important to rendering and scene browsing ([3, 7]), as well as to the representation conversion process.

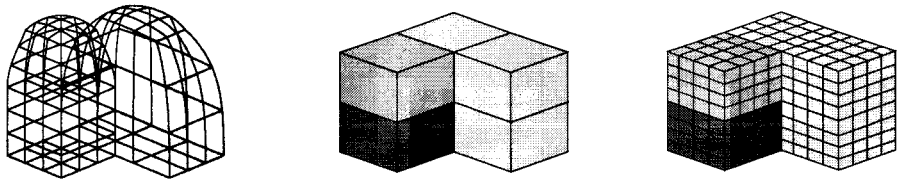
3 The Representation Conversion Process

One of the challenges of GGM is integrating interactive geometric modeling into the automated design process. To address this problem a generalized process for converting a geometric instance into a semantic description which can be understood by the generation system is required. This process is necessarily difficult since a geometric instance may have zero, one or many GGM representations. The conversion process is even more profound for a DGM system, since the generated representation must not only reproduce the original geometric instance, but also a set of instances which are in some way similar to the original. If such a conversion process can be devised, then not only can geometric instances be used to create IDR's, but the designer could also use the conversion process as a kind of visual programming. This new design description paradigm would allow the designer to describe a scene by creating a geometric example. The DGM system would then be responsible for generating a set of geometric instances which are similar to the example instance.

The algorithm presented here is tailored specifically for the MULTIFORMES DGM. Hierarchical decomposition requires that the scene be organized as a tree, and the MULTIFORMES generation system requires that each sub-scene, or node in the scene tree, be described by a set of rules. Thus the algorithm described generates a hierarchy of sub-scenes, then, based on their geometry, generates rules to describe each one. This basic strategy may be applicable to other GGM systems, but would have to be modified to produce scene descriptions with the appropriate structure and semantics.

The conversion algorithm attempts to reverse the top-down scene decomposition which MULTIFORMES uses in its scene description. By reversing this process, an entire scene can be built up by logically combining lower level elements. This bottom-up process uses three steps; scene element recognition and approximation, the agglomeration of connected scene elements and the positioning of disconnected elements.

In the first step, the geometric instance is examined and scene elements are identified. For example, the scene element used by MULTIFORMES is the compound primitive. Since a scene element may have been previously deformed, it may be in a form which MULTIFORMES can not describe in the IDR. For this reason, scene elements are approximated by their bounding boxes. Since the algorithm approximates sub-scenes by their bound-



(a) (b) (c)

Figure 3: A geometric instance (a) may be discretized by setting the size of one voxel to the size of the smallest possible compound primitive (b) or to an arbitrary size (c).

ing boxes, the conversion process is applicable to a wide variety of freely deformed input scenes, including those created with other geometric modelers. It may also be possible to re-insert the original, deformed sub-scenes after automated generation using the scene substitution process.

After the scene elements have been identified, they must be discretized and stored as a volume (voxel-based) representation. The voxel-based representation is used because it allows scene element connectivity determination to be performed using image processing techniques. Since MULTIFORMES controls the discretization of its design space, it may be possible to set the size of a voxel to be the size of the smallest possible undeformed compound primitive. This minimal discretization allows the scene to be represented in the smallest possible volume. Figure 3 shows a typical MULTIFORMES generated scene, (a), and how it can be approximated by a minimal volume, (b). Unfortunately, if no knowledge exists about the original discretization or the original scene was not discretized, an arbitrary discretization is required, such as that shown in figure 3 (c). After the discretization is complete, each scene element is described by generating a set of local inter-dimensional rules.

The second step of the conversion is to combine connected scene elements using rules of inter-scene position. The algorithm assumes that elements which are better connected have a more logical association that disconnected elements. The optimal agglomeration sequence can be constructed using the degree of connectedness as measure of the appropriateness of an agglomeration. For example, figure 4 shows two different sequences for the agglomeration of three scene elements. In sequence (a) the first agglomeration is partial, in that the two elements being combined do not share the entire face, while in sequence (b) they do. Thus, in the absence of any other information, sequence (b) is selected as the most appropriate.

Any simple or compound scene element is bounded by a set of planar faces. Using these bounding planar faces, two scene elements can be classified as being either fully connected, partially connected or disconnected. Two scene elements are said to be fully connected if they share a common face. If they partially share a common face they are partially connected and if they do not share a common face they are disconnected. Two scene elements which share a common edge are considered to be disconnected. Figure 5 illustrates these three levels of connectivity. This classification is best applied to scenes with a minimal volume representation, since otherwise most connected scene elements may be only partially connected.

After the connectivity of the scene elements has been determined, fully connected elements are combined. After all fully connected elements have been combined, partially connected elements are combined. Since joining two partially connected elements may produce a fully connected pair, a scene element constructed by combining two partially

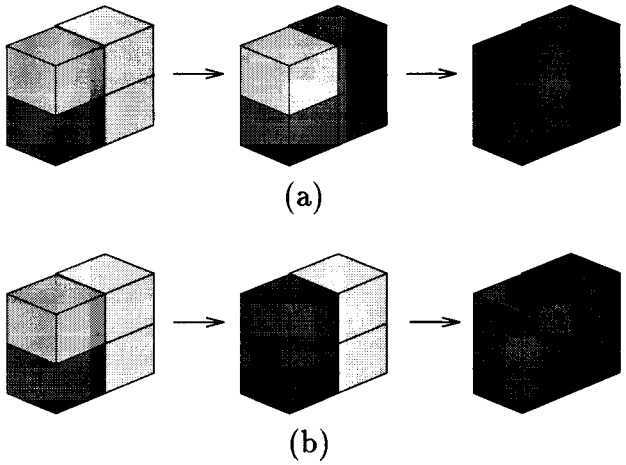


Figure 4: Two sequences for combining 3 scene elements; (a) shows a sub-optimal agglomeration which combines two partially connected elements before combining the fully connected ones, while (b) shows an optimal agglomeration sequence.

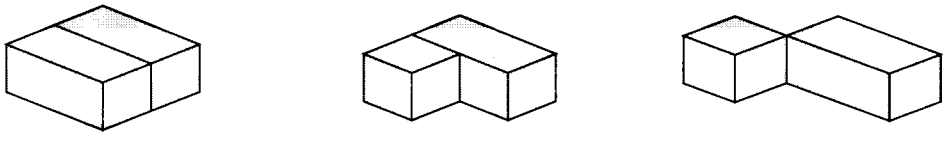


Figure 5: The three possible states of connectedness, fully connected (a), partially connected (b) and disconnected (c).

connected elements must be immediately tested for fully connected neighbors. If the newly created scene element has a fully connected neighbor, these two elements must be joined before any other partially connected elements are processed.

Elements are combined by creating a rule of inter-scene position which describes the agglomeration. This is followed by the creation of inter-dimensional rules describing the newly created scene element. This agglomeration algorithm is described in detail in [6].

The above technique assumes that scene elements which are *better* connected should be combined first (i.e. further down in the description tree). If a minimal volume representation does not exist, another technique for determining the best way to combine elements may be required, since in this case, most elements may be only partially connected.

Another approach to scene element agglomeration is to cast it as a combinatorial optimization problem. Using this approach, a measure of connectivity is used as an objective function, which is then maximized over a search space. In this case, the search space is defined as the set of all the ways in which a connected group of elements can be agglomerated. For example, if Ψ is the search space, with elements $\{\psi_1, \psi_2, \ldots, \psi_n\}$, and the element ψ_i is composed of the sequence of binary agglomerations, $\{\alpha_1^{x_1,y_1}, \alpha_2^{x_2,y_2}, \ldots, \alpha_m^{x_m,y_m}\}$, where agglomeration $\alpha_i^{x_i,y_i}$ represents the combination of scene elements x_i and y_i with the shared area of connection, κ_{x_i,y_i}, then the optimization seeks to maximize a objective function defining the total connectivity, K, such as,

$$K_\psi = \frac{1}{2m} \sum_{i=0}^{m} \left(\frac{\kappa_{x_i,y_i}}{A_{x_i}} + \frac{\kappa_{x_i,y_i}}{A_{y_i}} \right) \qquad \forall \psi \in \Psi,$$

where A_{x_i} and A_{y_i} are the areas of the partially shared face for scene elements x_i and y_i

respectively. This measure of total connectivity gives a value in $[0, 1]$ for all agglomerations in Ψ.

Even though the above technique may result in an agglomeration which is in some way optimal, solving the optimization may be difficult. For example, a similar combinatorial optimization problem has been shown to be NP hard ([1]). For large element agglomerations locally optimal solutions, generated using either greedy or heuristic techniques, may give acceptable results. It may also be possible to reduce the size of search space by culling of solutions which attempt to agglomerate widely separated elements early in the search tree. Another way to reduce the scope of this problem is by reducing the number scene elements to be agglomerated. If some knowledge exists about the logical organization of some sub-scenes, then element agglomeration can be performed as a preprocessing step, which would reduce the size of combinatorial optimization problem. For example, if a complex sub-scene has an existing declarative representation, and this sub-scene does not overlap with others, it may be treated as a single scene element.

The final step in the conversion algorithm is to combine disconnected scene elements. Since these elements are disconnected, the connectivity is not affected by the agglomeration sequence. That is, there is no sequence of agglomeration which will result in a fully or partially connected pair. For this reason, the logical association between disconnected elements is assumed to be weaker than that of connected elements, thus, such elements can be combined in an arbitrary sequence using inter-scene rules of position. Since there is not a strong logical association for the combined scene element, no inter-dimensional rules describing it are created.

There are three cases where the conversion process is required. First, a scene may have been created entirely with a TGM system, in which case there may be no available information about the logical organization of the scene. In this case, the automated process described above is required. However, if a scene was originally created by the automated system then some prior knowledge about the organization of the scene may exist. This information may be partial, for example, if scene inclusion rules were used, or incorrect, if the scene has been subjected to revision using the TGM system since it was created. In either case, the organizational information may be used to create a better reconstruction. A third case where a conversion process might be required is if design information is supplied directly from the designer using a visual programming environment. Such an environment could allow the designer to describe the size and position of scene elements, as well the the logical relationships between them.

Prior knowledge can be applied to the agglomeration problem in two ways. First, if a scene element is known to be an undeformed compound primitive, this information may be useful in determining a minimal volume representation. Second, scene elements, whose agglomeration sequence are known, can be treated non-decomposable units. This would reduce the number of scene elements to be agglomerated and could, if the known agglomeration sequence is correct, produce a more logical scene organization.

4 Conclusions

This work has focused on the development of interactive techniques for generative geometric modeling as part of the larger project to integrate the MULTIFORMES DGM system with a tailored TGM system. Specifically, this paper describes a technique to convert a geometric instance into a semantic representation which is understandable to the generation system. This conversion process is important, not only to support the iterative design

path required by the integrated system, but also to provide the basis for an interactive scene description paradigm based on visual programming by example.

The algorithm uses a three step process. First, geometric scene elements are approximated by their bounding boxes, then discretized and converted into a volume representation. The second step combines connected scene elements into a hierarchy which reverses the scene decomposition used by MULTIFORMES to describe complex scenes. Two approaches to this step were introduced. The first approach is based on combining fully connected elements before any partial connected elements are combined. The second approach is based on a combinatorial optimization which attempts to maximize the connectivity of combined elements. The final step in the algorithm combines disconnected scene elements to form a single hierarchical description for the entire scene.

If knowledge about the logical organization of the scene is available, either from the semantic or geometric representation, or directly from the designer, this information can be used by the agglomeration algorithm to produce a more logical scene description.

Several areas of research associated with this work still need to be addressed. First, several implementation strategies are being pursued to determine which technique is most appropriate. Specifically, it must be determined if the combinatorial optimization can be solved in polynomial time, or if some heuristic technique must be used to achieve an acceptable solution. Also, the design of an environment for visual programming is being developed to provide designers with a more appropriate description paradigm for geometric programming.

References

[1] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232, July 1995.

[2] J. Heisserman. Generative geometric design. *IEEE Computer Graphics and Applications*, pages 39–45, March 1994.

[3] S. Kockhar. CCAD: A paradigm for human - computer cooperation in design. *IEEE Computer Graphics and Applications*, pages 54–65, May 1994.

[4] A. Paoluzzi, V. Pascucci, and M. Vicentino. Geometric programming: A programming approach to geometric design. *ACM Transactions on Graphics*, 14(3):266–306, July 1995.

[5] D. Plemenos. Declarative modeling by hierarchical decomposition: The actual state of the MultiFormes project. In *GraphicCon '95*, St. Petersbourg (Russia), July 1995.

[6] D. M. Sellinger. The design of the XMultiFormes geometric modeling sub-systems. Rapport de recherche MSI 96-06, October 1996.

[7] D. M. Sellinger and D. Plemenos. Integrated geometric and declarative modeling using cooperative computer-aided design. In *Actes du Congres, Infographie Interactive et Intelligence Artificielle*, pages 135–147, Limoges (France), avril 1996.

[8] J. M. Snyder and J. T. Kajiya. Generative modeling: A symbolic system for geometric modeling. In *Computer Graphics Proceedings, Annual Conference Series*, pages 369–378. ACM SIGGRAPH, 1992.