Environment and Planning B: Planning and Design 2006, volume 33, pages 523 – 540

DOI:10.1068/b31150

Curve-based shape matching: supporting designers' hierarchies through parametric shape recognition of arbitrary geometry

Jay P McCormack
Design Advance Systems Inc., 40 24th Street, Pittsburgh, PA 15222, USA;
e-mail: jay@designadvance.com

Jonathan Cagan
Department of Mechanical Engineering, Carnegie Mellon University, Scaife Hall 214,
5000 Forbes Avenue, Pittsburgh, PA 15213, USA; e-mail: cagan@cmu.edu
Received 7 December 2004; in revised form 10 August 2005

Abstract. In this paper we introduce a subshape matching technique for shapes composed of curved lines that lie in a plane. An interpreter with this ability allows for the practical implementation of general shape grammars as well as opening the door for new applications of shape grammars. This method for matching shapes consisting of curved lines is outlined and examples are shown. The emergent properties of shape grammars are also explored and considered using the new method for shape interpretation.

1 Introduction

In our paper, "Supporting designer's hierarchies through parametric shape recognition" (McCormack and Cagan, 2002), we introduced a method for shape matching which allowed the designer to have more flexibility and control in specifying the range and type of valid matches in the implementation of straight-line shape grammars. This flexibility was necessary in order to permit designers to craft and implement shape-grammar rule sets that described a variety of products and devices. As researchers gain familiarity with the mechanics of shape computation, the shape grammars produced increase in complexity and elegance. Increased complexities come in the form of details represented in a grammar, and in the decreased limitations on the geometry defining a grammar. The maximum potential of a shape grammar is realized when the grammar is implemented, thereby allowing a user or artificial-intelligence-based tool to work interactively, generating designs in the language defined by the rules. Implementation requires a shape-grammar interpreter capable of performing subshape matching within more complex geometry, in order to determine in which case a rule can be applied. Subshape matching is also necessary to support the emergence of shape, the creative potential in shape grammars.

Previous shape-grammar authors often avoided the use of curves when defining shapes because of the issues involved with implementation, and instead used straight-line approximations. Grammars that used curves avoided shape-matching entirely, such as the coffee-maker grammar, which relied on labels. In this paper we outline a method for implementing shape grammars which are defined by curved lines in a plane based on our previous interpreter (McCormack and Cagan, 2002). Although presented and implemented in two dimensions the approach is equally valid in three dimensions. The method outlined for shape matching was implemented and used with the Buick brand shape grammar, as a grammar to design headlights for a vehicle.

We begin by reviewing our method for parametric shape matching through decomposition, then demonstrate how this method corresponds to a parameterization of a shape and the transformation of the right-side shape of a rule.

2 Shape grammars

Shape grammars (Stiny, 1980) are composed of shape rules which can be applied in a sequence, beginning with an initial shape, to create shapes in the language defined by the grammar. Shape rules take the form $a \rightarrow b$ where a and b are shapes. In order to apply this rule to shape c, an instance of a must be located in c. If a transformation of a, $\mathrm{t}(a)$, is located in shape c, the rule can be applied to produce a new shape c' by using the equation:

$$c' = c - \mathrm{t}(a) + \mathrm{t}(b) .$$

From this equation, and the requirements to apply this equation, it is apparent that three operations are necessary to implement a parametric shape grammar: parametric subshape matching, shape addition, and shape subtraction. Subshape matching determines if and where a rule can be applied, and addition and subtraction are used to apply the rule. Previously created shape-grammar interpreters did not support both subshape matching and parametric shapes. Systems that could locate subshapes could find matches only of the entire shape (similar shapes), whereas systems that supported parametric shapes used label matching to determine rule applicability or did not recognize the results of the interaction between combined shapes. The methods for addition and subtraction of shapes consisting of straight lines has been well documented previously (Krishnamurti, 1980). Our method for parametric subshape recognition is detailed in a previous paper (McCormack and Cagan, 2002) and is outlined in section 3. The idea is to use knowledge of increasingly constrained geometric relations to decrease the search during shape matching. The extensions and additions to these operations necessary for implementing shapes consisting of curved-line segments are subsequently introduced in this paper.

3 Shape matching through shape decomposition

In our approach parametric subshape recognition is achieved through a decomposition of shapes into a hierarchy of subshapes ordered by their decreasing restrictions. Instances of each of the subshapes are individually located in the design shape and are then reconstructed to form an instance of the entire shape. The basis for the hierarchy of subshapes can be specified by the designer directly or through a parametrization of a shape, and constraints placed on the parameters. These hierarchies are defined to offer a complete representation, which accounts for all spatial relations between line segments, in particular those which lie in a plane. Though the focus here is in two dimensions, the approach is extendible to three dimensions.

The levels of the hierarchy are defined so that the most-constrained lines of a shape are those lines that the designer intended exactly. These most-constrained lines have specified parametric relations to other line segments and those relations, if altered, will compromise the designer's intentions. Conversely, the lowest level of the hierarchy, which contains the least-constrained line segments, only implies a specific connectivity between line segments, thereby implying that a more extensive search will be necessary. The hierarchy allows the designer's intent of what constitutes a shape match to be specified. It permits the specificity of an entire shape to be stated (a square is any square, a rectangle is any rectangle) and also allows a shape to have varying levels of specificity internally (adjacent square and rectangle represents any square adjacent to any rectangle).

A typical set of spatial relations is used to form the default hierarchy of shape decomposition. The goal is to capture the intent of the designer that created this shape and to find shape matches that coincide with the designer's intent. The lines in the shape that the designer specified exactly must be separated from the lines in the shape that were intended as a general scheme. Separating the parts of the shape on the basis of the designer's intent creates a hierarchy of subshapes that parametrically represents the whole shape.

Of the possible transformations that could be applied to a shape, some will not destroy certain features and some will. For example, no amount of translation or rotation will destroy a specific feature, but, rather, will move the features from place to place, shearing, however, will eliminate any perpendicular intersections. Anisotropic scaling (different scaling factor in all directions) will destroy symmetry unless the scaling is along or perpendicular to the line of symmetry. Isotropic scaling (same scaling factor in all directions) and reflection (scaling by a factor of -1) does not, however, affect the symmetry of a shape. From this simple example groupings can be formed. The set of translation, rotation, and isotropic scaling will preserve all symmetry and perpendicular intersections. Translation, rotation, and anisotropic scaling will preserve the perpendicular intersections and symmetry in shapes that have parallel lines of symmetry or only one line of symmetry, if the axes of scaling are chosen properly. Translation, rotation, anisotropic scaling, and shearing can be applied to any intersecting line pair.

This leads to a four-level hierarchy or decomposition into subshape groupings as shown in table 1, which constitutes the default case. The default hierarchy is based on the assumption that the designer will specify a rule in such a way that the relations between line segments present in the rule are required. Upon decomposition, line segments will be placed into the most highly constrained subshape grouping for which they qualify.

Table 1. The four subshape groupings of the default hierarchy.

Subshape grouping	Features	Transformation
s_1	(1) lines that intersect perpendicularly and are the same length, (2) lines that are symmetric to multiple lines that are nonparallel.	translation, rotation, isotropic scaling
s_2	(1) lines that intersect perpendicularly, lines that are symmetric (2) to one line, or (3) to multiple lines that are parallel.	translation, rotation, anisotropic scaling
s_3	(1) intersecting lines.	translation, rotation anisotropic scaling, shearing
s_4	none	none

Grouping s_1 consists of the most highly constrained line segments that intersect perpendicularly and are the same length. Additionally, the s_1 grouping also contains any line segment that is symmetric to two or more other line segments, with the line segments nonparallel. Grouping s_2 contains the line segments that intersect perpendicularly. Any line segment that is symmetric to another line segment is also included in grouping s_2, including line segments that do not intersect. Grouping s_1 is a subset of s_2. Grouping s_3 contains the line segments that intersect. This makes groupings s_1 and s_2 subsets of s_3. The line segments in grouping s_4 have no discernable spatial relation to any other line segments, essentially those lines not found in s_1, s_2, or s_3. In these cases endpoints can be used as distinct points in order to reduce the number of shape matches to a finite value. This assumption is made in practice for any indeterminate situations and is often supplemented by labels to reduce the search. The s_4 line segments feature no actual line-segment intersections (unlike s_3) and hence are less specific and less deserving of distinct-point assumptions. A single-line segment in the s_4 grouping requires additional information and assumptions to be made in order to find a match. It is likely that the use of an s_4 line as a left-side shape in a rule would also include labels that give the line meaning.

Having established the set of parametric features that define the default hierarchy, it should be restated that the spatial relationships that define each part of the hierarchy of shapes can be customized by the designer to reflect his or her need. An alternative and complete decomposition technique is discussed in McCormack (2003). The default method of shape decomposition provides a complete representation of shape and has now enabled a shape to be divided into pieces, thus turning a difficult parametric-subshape-recognition process into a manageable task. The subshape pieces of a whole shape can now be dealt with individually when performing the subshape-recognition process and yet can allow an entire shape to be parametrically recognized through their combination. More details of decomposition with the use of the default hierarchy can be found in a previous paper (McCormack and Cagan, 2002).

The presentation of the hierarchical decomposition is done in a manner which implies that the shapes which define rules are decomposed in an effort to determine the intentions of the rule writer. In fact the rule writer has knowledge of and even determines the hierarchy while writing the grammar rules, and is therefore allowed to specify an entire shape and its decomposition on the basis of that hierarchy of spatial relations. This approach is explored next.

4 Two-dimensional parametric curved-line interpreter

An extension can be made to the hierarchical method of parametric shape matching of straight lines, which can be used for parametric recognition of curved-line shapes. This method uses a two-step approach that first finds a set of potential matches on the basis of a representative straight-line shape, and then validates or rejects the potential matches by comparing actual curves. This approach has advantages over the pure matching of characteristic polygons in that it can match equivalent curves with differing characteristic polygons, in addition to emergent shapes.

The derived straight-line shape is referred to as the distinct shape, as it is a collection of lines connecting distinct points from the curvey shape. The set of distinct points includes intersections between curves, curve endpoints, and projected intersections of well-known curves (such as circles). As was done in the straight-line case, endpoints are used as distinct points for matching otherwise indeterminate shapes. Figure 1 is a shape consisting of three curved lines in which the distinct points are marked. The connectivity of the distinct points is then represented by a set of straight-line segments, which correspond to a curved line or a portion of a curved line, and combine to form the distinct shape. Figure 2 shows the curved-line shape of figure 1, its distinct shape, and the correspondence between distinct lines and curved lines.

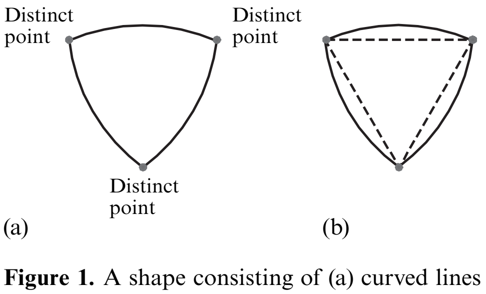

Figure 1. A shape consisting of (a) curved lines (solid) with distinct points marked and (b) curved-line shape (solid) and corresponding distinct lines (dashed).

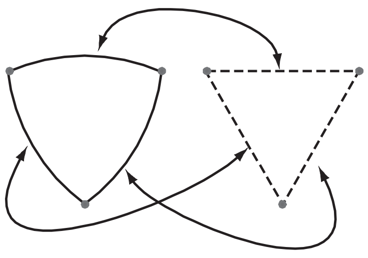

Figure 2. Curved-line shape (solid) and its corresponding distinct lines (dashed).

The distinct lines in figures 1 and 2 are shown as dashed lines, and the curved lines are solid. This convention is used throughout for clarity.

A hierarchy is again established that specifies the spatial relations and appropriate transformations at each of its levels. However, this method differs in that the transformations used during the matching of the distinct shape are derived from the spatial relations of the curved shape (that is, the symmetry of the shape requires the use of translation, rotation, and isotropic scaling, as shown in table 1) and not the distinct shape itself. This is done because, though the distinct shape serves as an equivalent of the curved shape for the initial coarse matching, it may misrepresent the spatial relations between the curved lines. Figure 3 shows two shapes consisting of curved lines and their distinct shapes, shape *p* and shape *q*. Shape *p* has three lines of reflective symmetry whereas shape *q* has only one line of reflective symmetry. However, they share the same distinct shape, an equilateral triangle. If transformations for matching were chosen on the basis of the distinct shape, a search for shape *q* may result in an incorrect set of matches. Hence, transformations for the coarse matching are derived from the curved-line shape. This is shown in a sample hierarchy in table 2. The two-stage matching

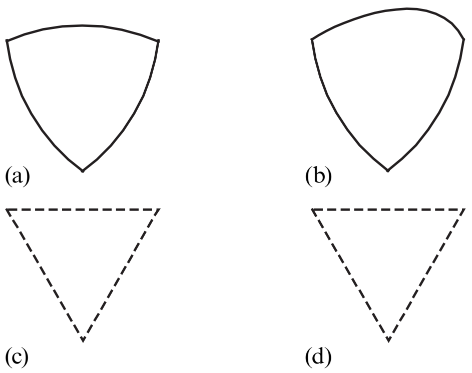

Figure 3. Two curved-line shapes and their corresponding distinct shapes demonstrate the loss of symmetry in the distinct representation—(a) shape *p*, (b) shape *q*, (c) district shape of *p*, and (d) distinct shape of *q*.

Table 2. Transformations for a distinct shape are derived from a curved shape.

Subshape group	Spatial relations	Transformations
(a) Curved shape		
s_1	symmetry across multiple axes that are not parallel	translation, rotation, isotropic scaling
s_2	symmetry across one axis	translation, rotation, anisotropic scaling
s_3	intersecting lines	translation, rotation, anisotropic scaling, shearing
s_4	the remaining lines	all
(b) Distinct shape		
s_1		translation, rotation, isotropic scaling
s_2		translation, rotation, anisotropic scaling
s_3		translation, rotation, anisotropic scaling, shearing
s_4		all

process is demonstrated through an example, and then a general approach is given as pseudocode.

Shape a is to be searched for in shape c_0 (figure 4). Shape a can be decomposed into a set of n subshapes and searched for individually, with the connectivity of the subshapes maintained throughout the search with labels, as was previously outlined for straight lines. In this example shape a has two subshapes, a_1 and a_2 (figure 5). The search begins by finding this distinct shapes d_i for each of the subshapes a_i and

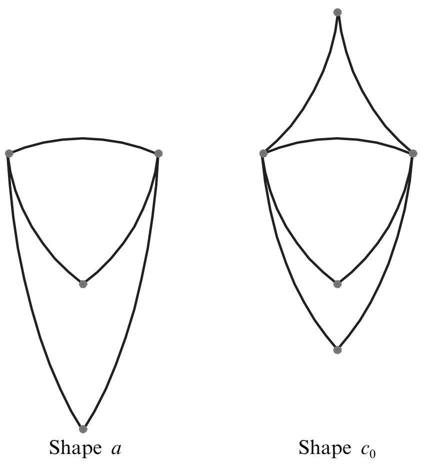

Figure 4. Instances of shape a are to be found in shape c_0.

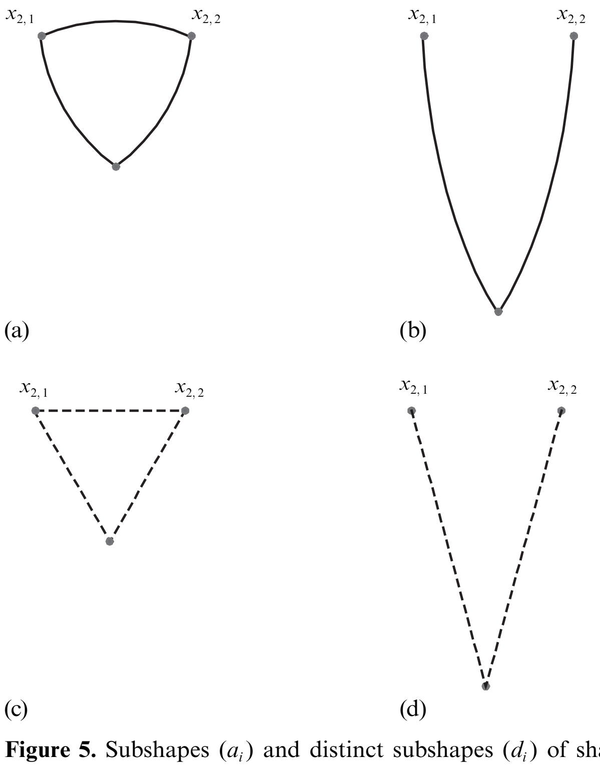

Figure 5. Subshapes (a_i) and distinct subshapes (d_i) of shape a—(a) subshape a_1, (b) subshape a_2, (c) distinct subshape d_1, and (d) distinct subshape d_2.

the distinct shape v_0 of shape c_0. For this search d_1 and d_2, the distinct shapes for subshapes a_1 and a_2, respectively, are shown in figure 5 and the distinct shape v_0 of shape c_0 is shown in figure 6. Shape v_0 is placed in the set of shapes V_0. This grouping is useful when the matching process is described as a loop.

Beginning with the most highly constrained level of the hierarchy, which is d_1 in this example, a search is performed for instances of d_i in each shape $v \in \{\mathrm{V}_{i-1}\}$,

$$\mathrm{F}_i = \{\tau(d_i) \leqslant v | \forall \tau \in \tau_i\} .$$

The set of transformations, τ, used in this search comes from those specified by the user in the hierarchy. In this example, shape d_1 is searched for in shape v_0 (the only shape in V_0) using the set of transformations τ_1 that are indicated as appropriate for a_1 by the user in the hierarchy. The hierarchy of spatial relations and transformations for this problem is shown in table 3. The search reveals two matching shapes with six different orientations of labeled points each. The two matching locations (without labeled orientation) are shown in bold in figure 7 (over). The search can be restricted further by noting that the labeled distinct points of shape d_1 must be matched with distinct points in v_0, each of which is the intersecting point of more than two line segments. The additional information reduced the number of matches to a total of four, which are stored in the set of shapes F_1 and are shown in bold in figure 8 (over).

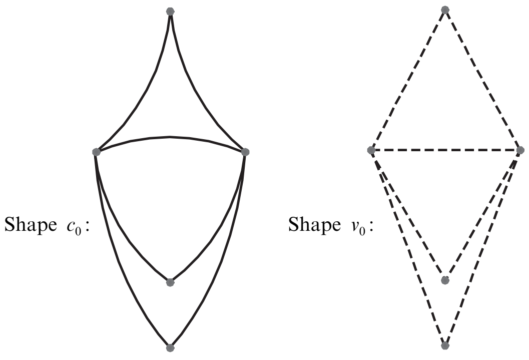

Figure 6. Shape c_0 and its distinct shape v_0.

Table 3. Hierarchy of spatial relations for shape a (figure 5).

Subshape group	Spatial relations	Transformations
(a) Curved shape		
s_1	reflective symmetry with more than one line	translation, rotation, isotropic scaling
s_2	reflective symmetry with one line	translation, rotation, anisotropic scaling
s_3	null	
s_4	null	
(b) Distinct shape		
s_1		translation, rotation, isotropic scaling
s_2		translation, rotation, positive anisotropic scaling
s_3	null	
s_4	null	

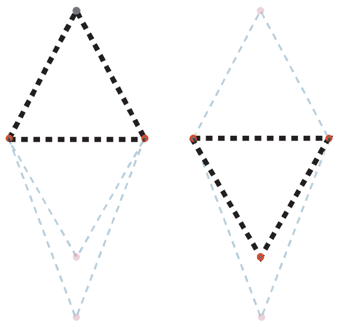

Figure 7. Distinct shape matches of subshape a_1 in v_0 shown in bold.

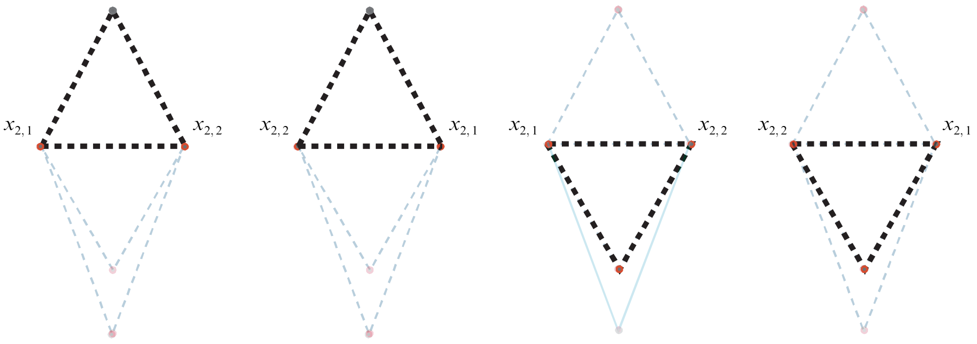

Figure 8. The set of shapes F_1.

Each shape f in the set F_i (instances of d_i in V_{i-1}) corresponds to a set of curved lines in shape c_0, which collectively form the curved-line shape f'. The corresponding curved lines, f', are compared with subshape $\tau(a_i)$, where the transformation τ is the same set of transformations used to match d_i. This is the final stage of matching. For each successful fine match f', the corresponding distinct shape f is placed in the set S_i. Returning to the example, the curved-line shapes corresponding to the distinct shapes in F_1 are compared with $\tau(a_1)$ in figure 9. There are two transformations of a_1 that past the fine check, and the corresponding distinct shapes are placed in the set S_1 (figure 10).

The fine check can be performed using a number of different methods. Control polygons can be compared for equivalence if the number of control points defining each curve is the same. Control points can be added to a polygon with fewer points without changing the curve, in order to allow for comparison. The addition of points and comparison of control polygons are simple and efficient processes. If the subtraction algorithm is efficient then the fine check can be performed by testing if $\tau(a) - c = \emptyset$. This will be true when all line segments in the transformed subshape a, $[\tau(a)]$, are present in c and are therefore a match. This is effective in that it uses an operation that was necessary for implementing the shape grammar.

Each shape s in S_i is subtracted from shape v in V_{k-1} for which it is a subshape and the share points between the difference and s are labeled to maintain connectivity,

$$\forall s \in S_i,\ V_i = \{g(s, v - s)\}\,,$$

where $g(y, u)$ is the operation that transfers the connectivity labels from shape y to shape u. The labels are of the form $x_{j,m}$, where j is the less-constrained subshape grouping number and m is an index. This is the same labeling technique described

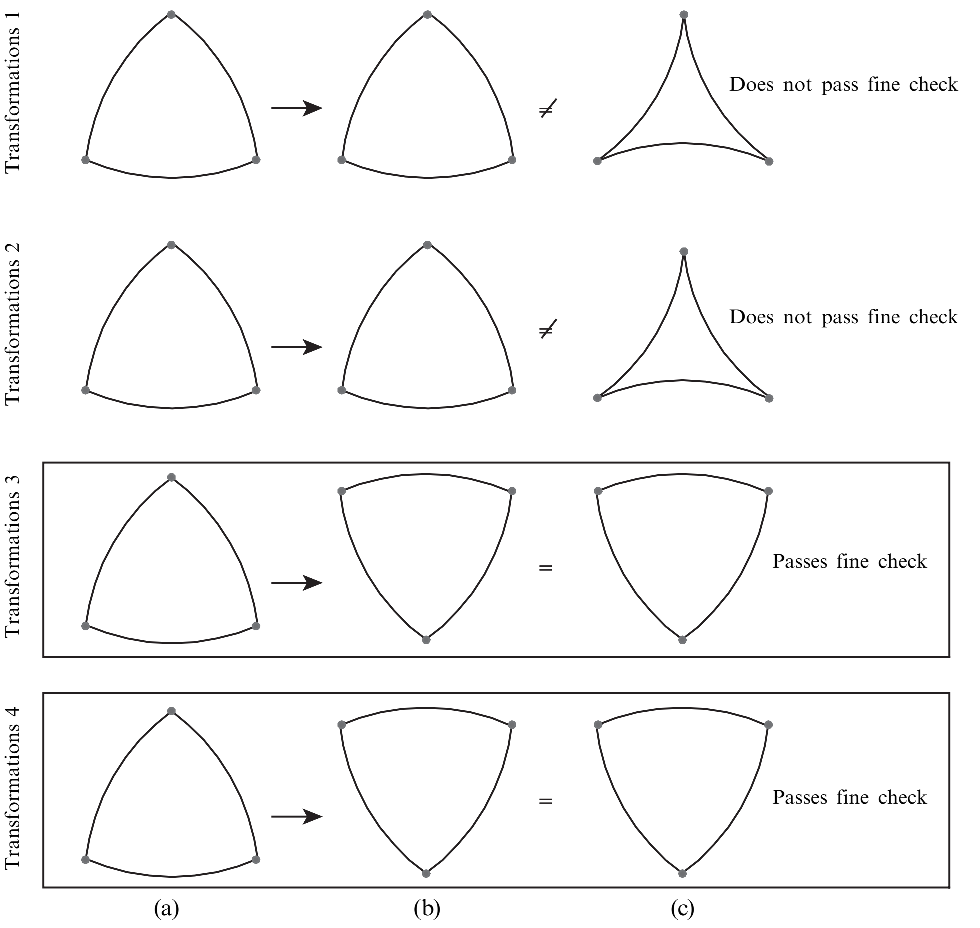

Figure 9. (a) a_1, (b) transformed a_1, (c) f', the curved lines corresponding to matches in F_1.

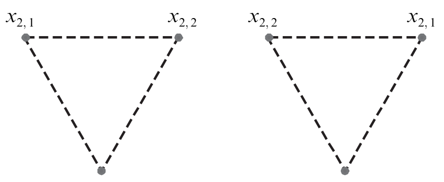

Figure 10. The set of shapes S_1.

for straight-line matching. The resulting labeled shapes are stored in the set V_i. In the example each shape in S_1 is subtracted from shape v_0 (the only shape in V_0) and the shared points between the difference and the shapes in S_1 are labeled as such. The resulting labeled shapes are stored in the set V_1 (figure 11, over).

The general method is used iteratively until all subshapes of a have been searched for or until one of the searches produces no results. If one of the subshape searches is unsuccessful then there is no instance of shape a in c_0. If the search is successful for each subshape then instances of shape a in c_0 are produced by subtracting each shape in V_n from v_0 and finding the corresponding curved lines of the difference.

The shape in the example continues with a search for the distinct shape d_2 in each shape in V_1 through the use of the appropriate transformations from the given hierarchy.

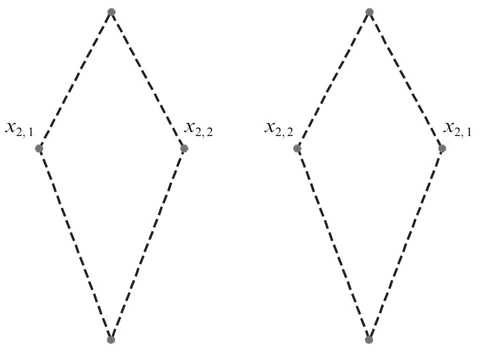

Figure 11. The set of shapes V_1.

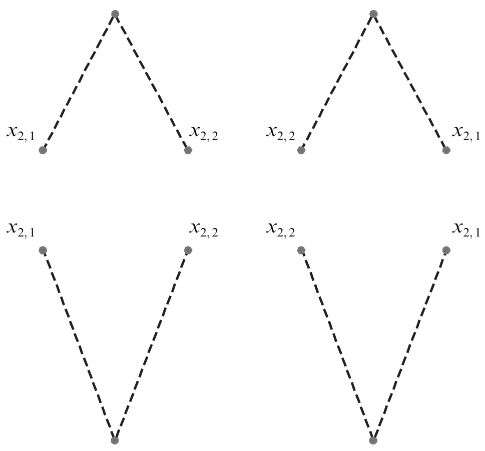

Figure 12. The set of shapes F_2.

There are four matches, which constitute the set of shapes F_2 (figure 12). The curved-line shapes corresponding to the distinct shapes in F_2 are then compared with a transformed version of a_2 (figure 13). Here two of these shapes pass the fine check and the corresponding distinct shapes are placed in the set of shapes S_2 (figure 14). Each shape in S_2 is then subtracted from the shape in V_1 of which it is a subshape. The resulting difference of shapes is stored in the set V_2 (figure 15, over).

There are no additional subshapes of shape a so the search is terminated. The matching instances of shape a in c_0 are the curved-line shapes corresponding to the difference of distinct shape v_0 and each shape in V_2. The result of subtracting each member of V_2 from v_0 is shown in figure 16 (over). The corresponding curved-line shapes and matches of a in c_0 are shown in figure 17 (over). The general curved-line matching process is described in pseudocode form in figure 18 (over).

This combination of matching steps is capable of finding emergent shapes. The straight-line matching procedure is able to find emergent shapes (McCormack and Cagan, 2002), and relies only on maximal line segments as a required input. This approach derives the straight-line shape from the curved shape to produce a set of maximal line segments, and enables the straight-line matching to find any emergent shapes. The remaining role of the curve-matching procedure is to resolve differences in the curved-shape representations; this is accomplished with the use of standard geometric algorithms (that is, De Casteljau's algorithm). However, it has not been proven that the translation from a curved-line representation to the straight-line representation is guaranteed to match every emerging shape.

5 Sample rule application

Through the use of the matching, addition, and subtraction operations defined for curved lines, shape rules consisting of curves can be applied. A one-rule grammar is defined by the rule in figure 19 (over) and the initial shape in figure 20 (over). In order to apply this rule to the initial shape, an instance of the left-side rule of the shape must be located in the initial shape.

The matching process begins by forming the distinct shapes of the left-side shape and the initial shape. The left-side shape and its corresponding distinct shape are shown in bold in figure 21, and the initial shape and its distinct shape are shown in figure 22. In figures 21 and 22, the actual curved lines are shown with solid lines, and the distinct lines are represented with dashed lines.

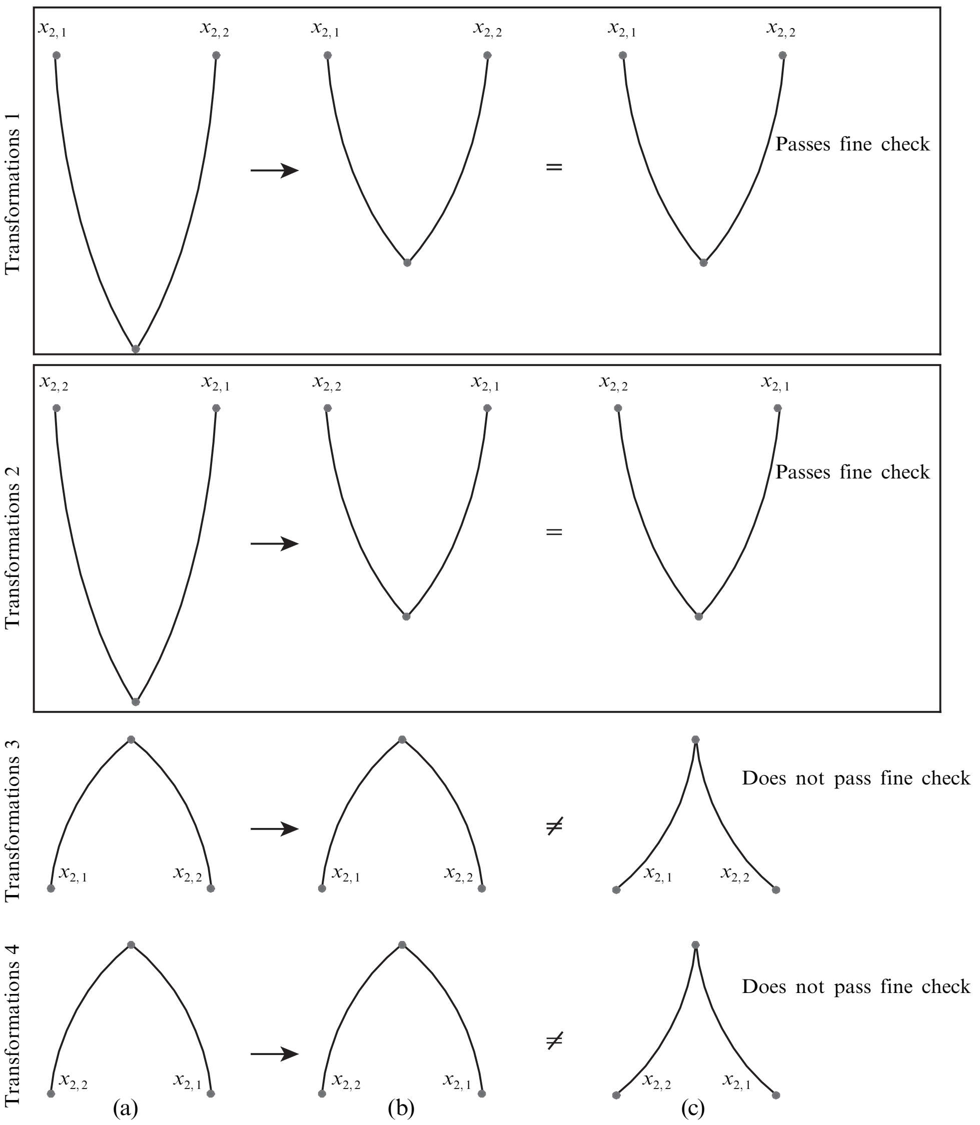

Figure 13. (a) a_2, (b) transformed a_2, (c) curved lines corresponding to matches in F_2.

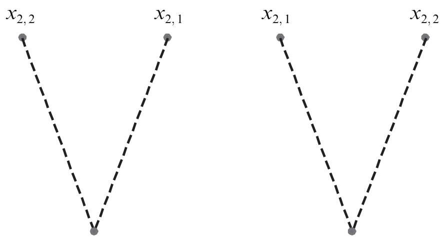

Figure 14. The set of shapes S_2.

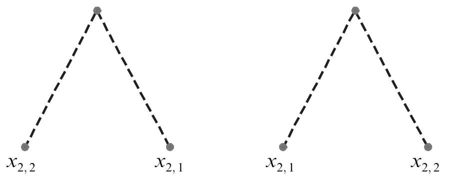

Figure 15. The set of shapes V_2.

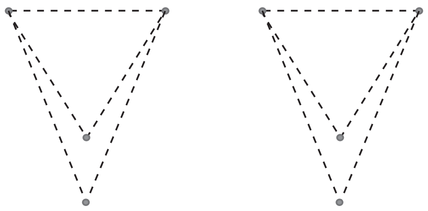

Figure 16. Distinct shape matches of d in v_0.

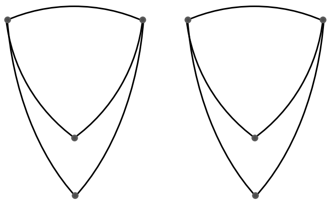

Figure 17. Instances of shape a in shape c_0.

```
Notes for pseudocode:
Lower case represents a single shape
UPPER CASE represents a set of shapes

Given shape $a$, find an instance of shape $a$ in shape $c_0$ by using a user-defined shape hierarchy of $n$ levels

Determine shape $d$, the distinct shape of shape $a$
    Decompose shape $a$ into $n$ subshapes where $a_i$, $i = 1, ..., n$, corresponds to subshape grouping $s_i$ from
    the user's hierarchy and shape $d_i$ is the distinct shape corresponding to subshape $a_i$

Shape $c_0$ is placed in the set of shapes $C_0$
Shape $v_0$, the distinct shape of shape $c_0$, is placed in the set of shapes $V_0$

$m = 0$
For $i = 1$ to $n$
    For $j = i + 1$ to $n$
        Mark shared points between $d_j$ and $d_i$ with label $x_{j,m}$
        $m = m + 1$

For $k = 1$ to $n$
    For each shape $v$ that is a member of the set of shapes $V_{k-1}$
        Search for instances of shape $d_k$ in shape $v$ using $\tau_k$, the transformations specified
            As appropriate for $a_k$
        Store all instances of $d_k$ in $v$ whose labels at levels $1, ..., k$ match in $F_k$
            For each distinct shape $f$ that is a member of the set of distinct shapes $F_k$
                Shape $f'$ is the curved lines in $c_0$ corresponding to $f$
                If $f' = \tau_k(a_k)$
                    Put $f$ in set of shapes $S_k$
        For each shape $s$ that is a member of the set of shapes $S_k$
            $v' = v - s$
            Copy the shared point labels between $v'$ and $s$
            Put $v'$ in set of shapes $V_k$
            If $k = n$
                For each shape $v_k$ that is a member of the set of shapes $V_k$
                    An instance of the original shape is equal to the shape corresponding to the distinct
                    shape $v_0 - v_k$
```

Figure 18. Pseudocode describing curve-matching process.

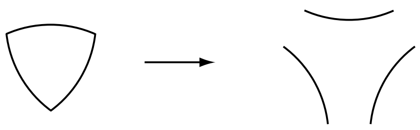
Figure 19. A shape rule based on curved-line segments.

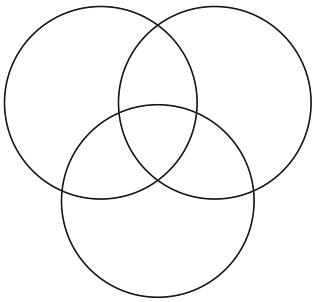
Figure 20. Initial shape.

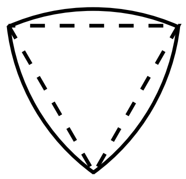
Figure 21. The left-side shape from the rule (solid line) and the corresponding distinct shape (dashed line).

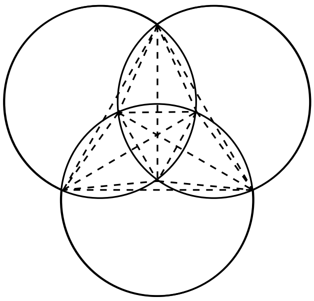
Figure 22. The initial shape (solid line) and the corresponding distinct shape (dashed line).

The symmetry between the curved lines of the left-side shape across multiple axes indicates that only similarity transformations (translation, rotation, and scaling) can be used in the search for the left-side distinct shape in the distinct shape of the initial shape. The selection of these transformations was discussed in section 3 and is summarized in table 1. Through the use of these transformations two distinct-shape matches are found and are highlighted in figure 23. The curved lines corresponding to the distinct matches are then examined in figure 24 (over), with only 24(b) validated as a curved match. The rule is then applied through the subtraction of the matching subshape (highlighted in figure 25, over), which results in a new shape, shown in figure 26 (over). The right side of the rule in figure 19 is then transformed with the use of the same values of rotation, transformation, and scaling which produced a match of the left-side shape. The transformed right-side shape is then added to the

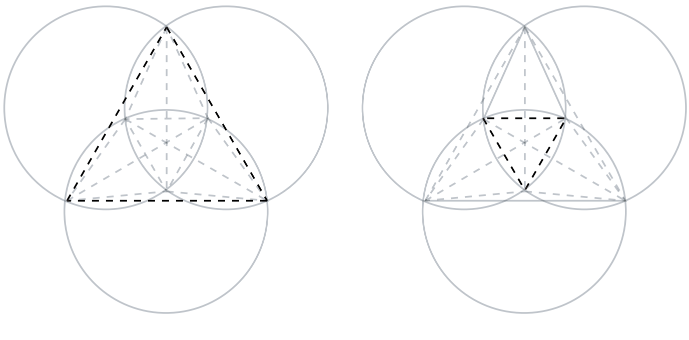
Figure 23. The distinct shape matches found in the initial shape.

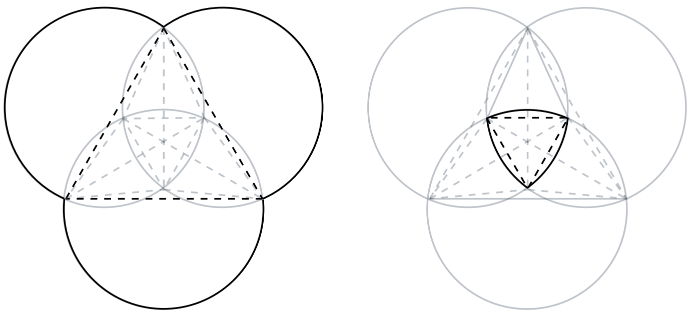

Figure 24. The curve segments corresponding to the distinct matches in the initial shape.

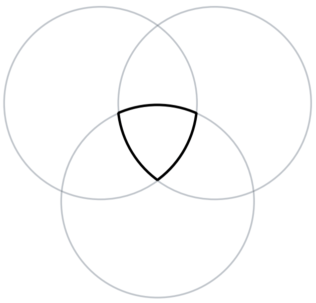

Figure 25. An instance of the left-side shape in figure 21, in the initial shape from figure 20.

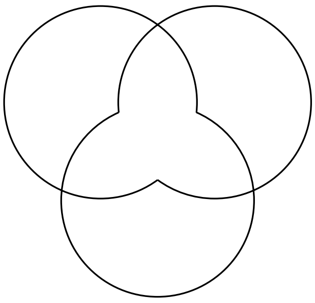

Figure 26. The difference of the initial shape and the instance of the left-side shape found in the initial shape.

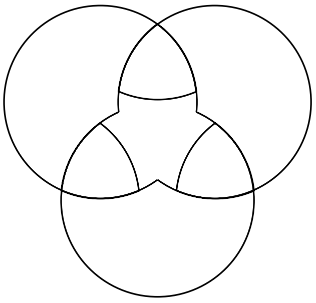

Figure 27. The shape resulting from the application of the rule in figure 21 to the shape in figure 20.

shape in figure 26, which completes the rule application and produces the new shape shown in figure 27. The rule from figure 19 can then be applied to the same shape four more times, thereby producing the language of shapes seen in figure 28.

6 Using the interpreter

The availability of a curve-based interpreter allows the general implementation of shape grammars that adhere to its geometric limitations (that is, curved and straight lines in a plane) and thus enables a designer to focus on defining rules, languages, and designs. The implementation process consists of defining the geometry of the rules (two shapes per rule), a decomposition of those shapes, an accompanying set of transformations, and an initial shape as a starting point.

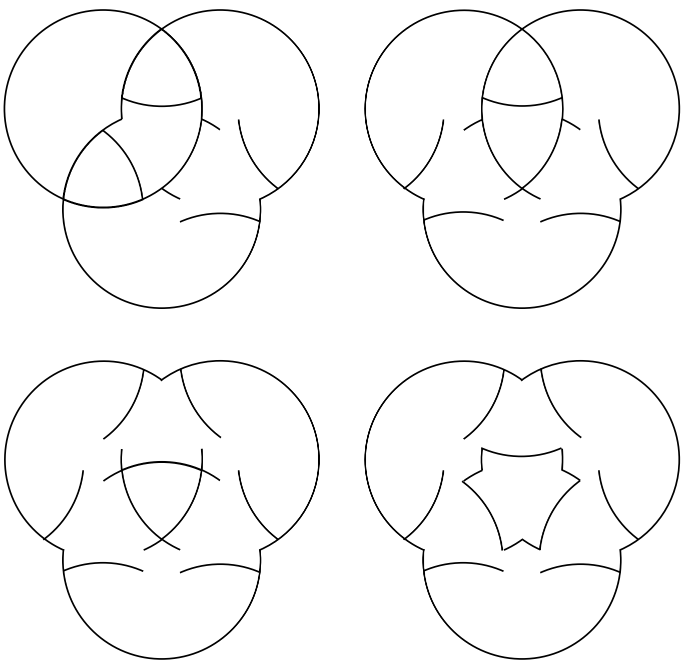

Figure 28. Shapes resulting from subsequent applications of the rule in figure 19 to the shape in figure 27.

The Buick brand shape grammar (McCormack et al, 2004), consisting of seventy rules, was implemented in a few hours time using the curved-shape interpreter with only a text-based interface. Working with the interpreter allows the designer to efficiently explore designs in the brand language and allows the language to be improved through rule additions and changes to the language. The user selects from a list of rules and the interpreter performs a search for the left-side shape of that rule. If a match is found the user is prompted to select which location in which to apply the rule and to fill in any open parameters that are part of the rule. Some examples of designs created with the implemented Buick brand grammar are shown in figure 29.

Another short grammar was written for use with the curved-line interpreter. This grammar was used as an experiment in user–grammar interaction to generate headlight designs as well as an exploration of defining rules from emerging shapes. Automotive headlight design is influenced heavily by technological advancements.

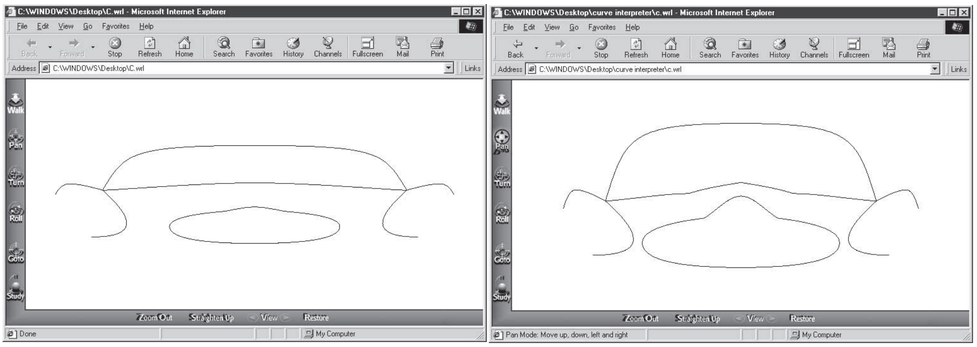
C:\WINDOWS\Desktop\C.wrl - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address C:\WINDOWS\Desktop\C.wrl
Links
Walk Pan Turn Roll Goto Study
Zoom Out Straighten Up View Restore
Done
My Computer
C:\WINDOWS\Desktop\curve interpreter\c.wrl - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address C:\WINDOWS\Desktop\curve interpreter\c.wrl
Links
Walk Pan Turn Roll Goto Study
Zoom Out Straighten Up View Restore
Pan Mode: Move up, down, left and right
My Computer

Figure 29. Some results of working with the implemented Buick grammar.

Recent trends show that the lights (headlights, turn signals) are becoming more three-dimensional objects, with their covers wrapping around to the sides and even the hood surface of the vehicle. This was made possibly by the molding of transparent plastics into three-dimensional shapes and made desirable by a trend in consumer products to feature transparent parts in their design (most notably the original iMac). The activity both of creating the rules and of exploring the language of shapes defined by those rules will be shown to provide inspiration for a new design.

Two rules were created (figure 30) and used to generate the shape in figure 31, from which the user selected an interesting region, which is indicated by the rectangle. Rotations used for matching were limited to 90° increments in order to eliminate indeterminacy. Through the choice of emergent shapes from figure 23, two new rules were created and are shown in figure 32. A new design (figure 33) was created with the use of four rules and with the shape in figure 31 as a starting point. From the new design (figure 33) an emergent shape is manually extracted, clipped, and detailed. The concept for a new headlight, parking light, and turn signal created with the grammar is presented as an alternative for the Buick Rendezvous in figure 34. These rules were implemented with our interpreter, which performed the shape computations. However, owing to the text-based interface, illustrations of the results were drawn by hand.

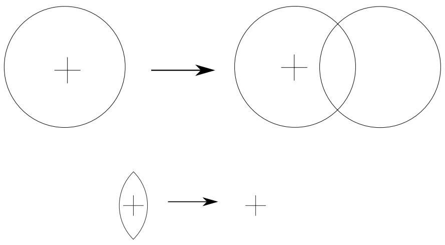
Figure 30. A two-rule grammar for generating headlight designs.

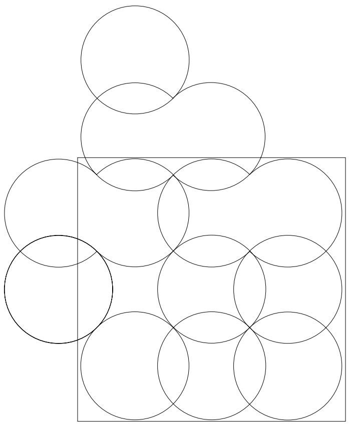
Figure 31. A shape in the language defined by the rules in figure 30. The selected area acts as a starting point for the next step.

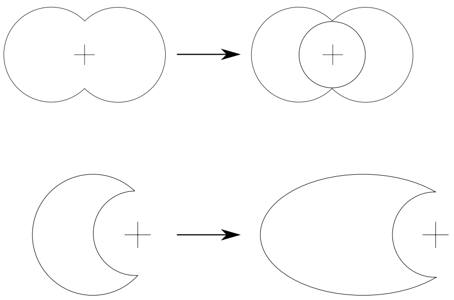
Figure 32. New rules added to the headlight grammar on the basis of emergent shapes in figure 31.

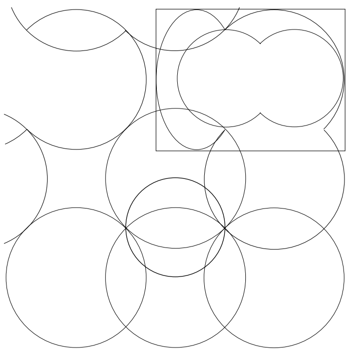
Figure 33. A shape in the language defined by the four rules of the headlight grammar.

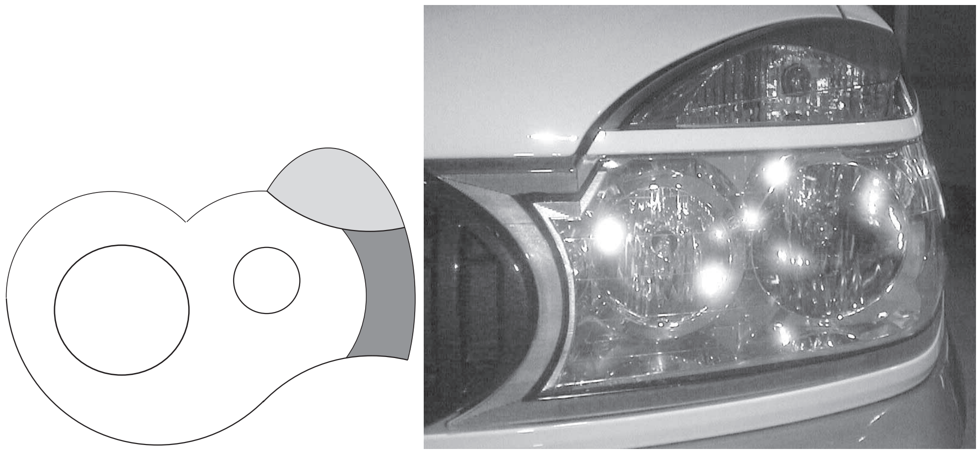

Figure 34. A headlight design that was extracted from the shape in figure 33 is presented as an alternative for the Buick Rendezvous.

7 Conclusions

This work introduced a curve-based shape-grammar interpreter using parametric subshape recognition. The interpreter has been implemented in C++ and was compiled and run on a Windows workstation. Unlike previous interpreters that are limited to nonparametric subshape matching or parametric shape matching with no emergence, or interpreters of only straight lines, our interpreter works on general parametric features through a decomposition technique which allows subshape matching. The method uses a two-step approach that first finds a set of potential matches on the basis of a representative straight-line shape and then validates or rejects the potential matches through a comparison of actual curves. The decomposition allows user intent to persevere in the shape-matching process.

Ordering the groupings of parametric relations into a hierarchy of decreasing specificity results in a more efficient subshape search order. Although in the worst case an exhaustive check of all distinct (that is, intersection and end) points is required, in practice labels restrict the number of points that must be examined for any shape-matching process. Other steps could be taken to improve performance, such as restructuring the subshape groupings.

Our parametric shape-grammar interpreter can be used to implement any created two-dimensional shape grammar which relies on matching to determine rule application and uses the shape-replacement rule $A \rightarrow B$. This interpreter was used to implement several grammars that contain, and rely on, curve-based shape emergence.

The extension to the parametric shape-grammar interpreter that allows for curved shapes is a valuable addition to shape-grammar implementations. The ability of the rule writer to use curves creates the possibility for more detail in representation and opens the door to the modeling of new types of products, which could not be adequately represented by straight lines. Brand modeling and aesthetic studies are examples of grammars in this category. Additional extensions to shape-grammar interpreters will further the potential for productive grammars and implemented grammar systems to emerge.

Shape emergence is a key difference that separates shape grammars from traditional expert systems. Having the technology to quickly implement shape grammars of arbitrary geometry allows the design process using shape grammars, including issues of emergence, to be studied through experiments and testing. By facilitating the creation of novel shapes, shape grammars can produce more than a predictable combination of parts, and implemented shape grammars can be the source for inspiration through the ambiguity of intersecting line segments.

References

Krishnamurti R, 1980, "The arithmetic of shapes" *Environment and Planning B* **7** 463 – 484

McCormack J P, 2003 *Implementing Parametric Shape Grammars to Capture and Explore Product Languages* PhD dissertation, Carnegie Mellon University, Pittsburgh, PA

McCormack J P, Cagan J, 2002, "Supporting designers' hierarchies through parametric shape recognition" *Environment and Planning B: Planning and Design* **29** 913 – 931

McCormack J P, Cagan J, Vogel C M, 2004, "Speaking the Buick language: capturing, understanding, and exploring brand identity with shape grammars" *Design Studies* **25** 1 – 29

Stiny G, 1980, "Introduction to shape and shape grammars" *Environment and Planning B:* **7** 343 – 351