Revisiting Shape Embedding

Hacer Yalım Keleş[1], Mine Özkar[2], Sibel Tarı[3]
[1,2,3]Middle East Technical University, Turkey
[1,2,3]http://www.metu.edu.tr
[1]hacerkeles@gmail.com, [2]ozkar@metu.edu.tr, [3]stari@metu.edu.tr

Abstract: *We propose and describe a working computer implementation for shape grammars that handles embedding relations in two dimensional shapes. The technical framework proposed explores a graph data structure to temporarily represent boundary elements of shapes and how they are assembled. With the associated algorithms, this structure enables a systematic search for parts. The employment of user defined constraints allows for an interactive search. In accordance with the continuous character of shapes, the study puts forth a practical part detection method, which extends to non-deterministic cases.*

Keywords: *Shape grammar interpreter; computation with shapes; part relations.*

The problem of shape representation

The theory of shape grammars frames a functionally varying perception of shapes as one of the key features in visual computing. Visual rules represent the desired action to be performed on a shape. A shape is not a predefined entity but is temporarily decomposed into parts that are useful in applying a given visual rule. Thus, the recognition of any relevant part embedded within a given shape of any kind without a predefined decomposition emerges as a fundamental technical problem to be resolved for visual computing.

Starting from the pioneering work of George Stiny and James Gips (1972), numerous significant works related to shape grammars and their implementations have made it possible for a broad research field to come to being. The recent reinstating of the philosophical and technical characteristics of the theory by Stiny (2006) has provided researchers in the field with a refreshed reference point and opportunity to evaluate the field.

However, many of these shape grammars are not computationally implemented while the existing shape grammar implementations are mostly application specific and provide partial solutions to the more general problem of computational shape embedding. Agarwal and Cagan (1998) and Pugliese and Cagan (2002) employ symbols in matching subshapes and focuses on specific engineering shape grammars.

Another prominent approach proposed for sub-shape recognition problem is the algorithmic representation of shape rule application developed by Krishnamurti (1980, 1981) based on maximal elements. Krishnamurti sets the basis for works that implement a shape grammar interpreter supporting emergent sub-shapes (Krishnamurti and Giraud,

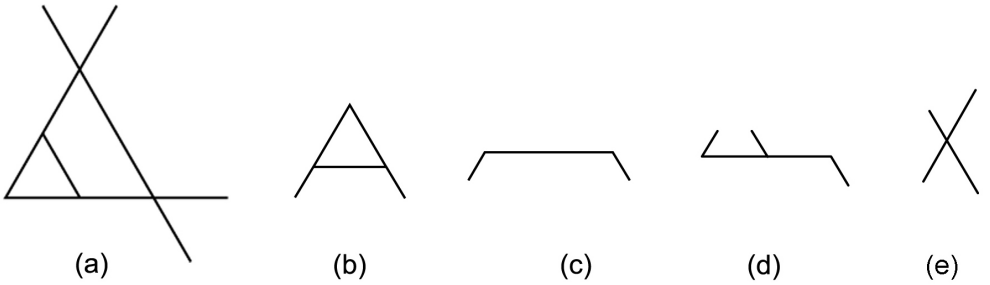


Figure 1
(a) An initial shape and (b-e) possible parts embedded in it

1986; Chase, 1989; Tapia, 1999).

This paper illustrates a sub-shape detection method that technically varies due to its flexible shape representation extending to non-deterministic cases. Without giving a set definition for the initial shape in Figure 1(a), we illustrate the detection of the four particular parts given in Figure 1(b-e) each of which represents a different technical problem.

Shapes, in this paper, are left true to their continuous nature and are considered with their boundaries as inherent properties of this nature. We assume a set of labeled points for each shape based on some topological descriptions that incorporate boundaries and boundaries of parts. Each description is a set of assumed primary features and how they are assembled. Any description, represented through labels, is arbitrary, disputable and temporary. The shape is left as you see it.

We propose for the given line shape(s) Figure 1(a) some topological descriptions that incorporate boundaries and boundaries of parts. Such a description is acquired through two classes of points that are (1) boundary points and (2) intersection points of maximal lines. This is similar to the decomposition points developed by Prats *et. al.* (2006) in their study on curved line shapes. However, while they introduce a four layer construction of contour, decomposition, structure and design for shape representation, we do not imply such constructions but, maintain that the intersection points are coincidentally boundary points of some embedded parts. We define these as points coincident with more than one maximal line in the shape.

Observing the intersection points and the boundary points of the four maximal lines in the given shape, we call these points as topologically critical points and identify some line segments that are bounded by these points. Topologically critical points are changeable.

Mapping a U_{12} shape to a V_{02} shape

Let us define a shape in U_{12} with a finite number of segments:

$$S_{U_{12}} = \{l_1, l_2, ..., l_n\} \quad (1)$$

In this representation, the definition of topologically critical points of a shape can now be unified to a class of boundary points of the said line segments. A shift from U_{12} to U_{02} can be defined as a mapping $M: U_{12} \rightarrow V_{02}$ for any shape using the boundary relation for each line segment designated in that shape.

$$M(S_{U_{12}}) = \{(p_{11}, L(p_{11})), (p_{12}, L(p_{12})), (p_{21}, L(p_{21})), ..., (p_{n1}, L(p_{n1})), (p_{n2}, L(p_{n2}))\} \quad (2)$$

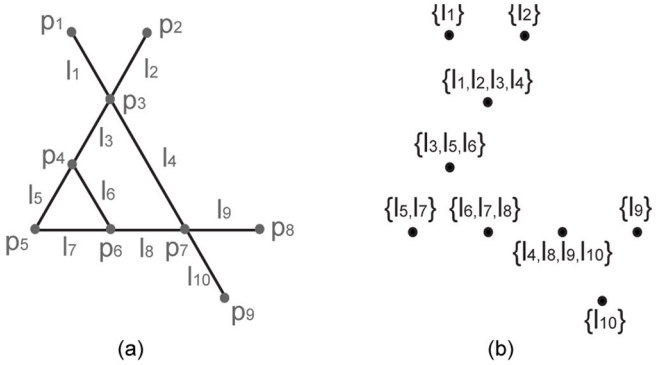


Figure 2
(a) Shape in U_{12}.U_{02} with names of line segments and points shown for illustrative purpose
(b) Shape in V_{02}

p_{i1} and p_{i2} are the boundary points of the line segment l_i and L*(p)* gives the label(s) of point *p*. The labels encode the relations between the points in U_{02} and refer to the line segments in U_{12}. For each point, a set of line segments that the point is coincident with is stored in a list of labels as shown in Figure 2.

Attributed representation of shapes

For a computer implementation, we represent these points and their relations in label algebra as an attributed undirected graph. This graph is a temporary data structure. The vertices of the graph refer to points and the edges of the graph refer to the relations between these, stored in the labels.

When a shape in U_{12} is mapped onto a shape in V_{02}, it is in fact mapped onto an attributed undirected graph $G = (N, E)$, where N is the set of points (nodes) of the shape in V_{02} and E is a Boolean relation such that there is an edge from node n_i to n_j if $(n_i, n_j) \in E$. E is defined below:

$$\text{if } \mathrm{L}(n_i) \cap \mathrm{L}(n_j) \neq \phi \text{ then } (n_i, n_j) \in E \quad (3)$$

Labels stand for two kinds of attributes for each node. One of these is the position of the node in terms of the local image coordinates of the points. The other is the information of how the points bound parts of the shape. This information varies according to the user and the type of shapes handled. For example for line shapes, for efficiency considerations for our algorithms, we store node-to-node connection angles.

This mapping readily applies not only to linear shapes but to curved shapes as well. The edges of the graph refer to arbitrary segments induced by the shape. For instance the wiggly versions of the shapes shown in Figure 1 can be mapped.

Identity and part relations of shapes

The mapping of the shape to a graph is not a one-to-one relation where each shape is uniquely represented with a graph and where each graph uniquely maps to a shape. The graph representations show the assumed topologically critical points and relations between these points. For a given shape, the relations between the points in the graph representations do not change when shapes go under Euclidean transformations. Only, the positions of the nodes change. For seeking an identity relation between two line shapes, we define a constrained structural isomorphism between their representative graphs to preserve edge connection angles and edge length ratios in addition to the adjacency relations.

A shape can be perceived in different instances where its scale, orientation and position change. All these instances are identical shapes under Euclidean transformations. We seek correlation between the graph representations of shapes in order to detect identity embedding relations between shapes.

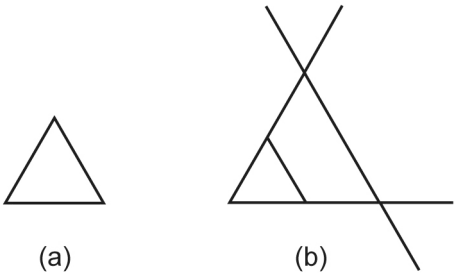


Figure 3
Sample shape (a) to be detected in shape (b)

Let us assume that we are to detect shape (a) as a part of shape (b) in Figure 3. There are more than one occurrences of (a) within (b) in different scales and orientations. When the simple graph of (b) is traversed to obtain the sub-shapes which are shape equivalent to (a), we obtain two simple subgraphs for which nodes and node connections are shown in Figure 4(a,b). Although both are shape equivalent to Figure 3(a) under some Euclidean transformations, Figure 4(b) is not graph equivalent to Figure 3(a) as it has more nodes than the boundaries of its maximal lines.

Instead, the corresponding subgraph representation of Figure 3(b) should be as in Figure 4(c) where the only nodes that define the subgraph are topologically critical points.

For cases such as these, where the graph representation for the embedded part is not completely a subgraph of the graph representation of the shape it is a part of, we extend the definition of a simple graph to obtain the over complete graph representation, so that for every possible traversal of the target graph there exists a subgraph which is shape equivalent to the traversed subgraph and the nodes of which conform to be topologically critical. This is achieved by using the relations between the points, which have implicit connections in addition to direct (explicit) connections.

As seen in Figure 4(b), the criticality of a point depends on the context. These points are boundaries of other shapes embedded in the initial shape. In the case of linear shapes, for all the points, which are coincident with the same straight line, there is an implicit connection between each pair of points when the shape is considered. By means of what we will hereon call an over complete graph representation of a shape, all implicit connections are made explicit, and the subgraph shown in Figure 4(c) become a valid subgraph of the graph in Figure 3(b).

Let us call a graph $G' = (N', E')$ an over complete graph where N' is the accepted set of points of the shape in V_{02} and E' is a Boolean relation such that there is an edge from n_i to n_j if n_i and n_j are both coincident with the same straight line. This relation can be obtained by examining the connection angles. For example, assume that node n_i has a connection with a node n_k with a connection angle α, and node n_k has a connection with another node n_j with the same connection angle (Figure 5). Then n_i is connected to both n_k and n_j.

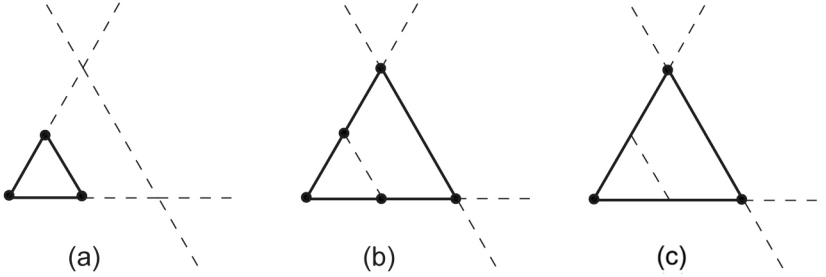


Figure 4
(a-b) Subgraphs for the graph of the shape in Figure 3 (b)
(c) A subgraph of the graph of the shape in Figure 3 (b)

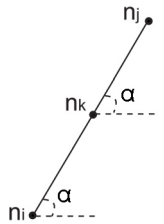


Figure 5
Node connection angles between node pairs (ni, nk) and (nk, nj)

Connection angles between nodes in a simple graph facilitate the identification of a set of nodes, which are coincident with the same straight line. In an over-complete graph representation, all of the nodes in this set are shown to have a direct connection with each other. It is thus another representation of the given shape in V_{02}.

As the definition of this extension implies, a simple graph $G = (N, E)$ of a shape *S* is a subgraph of an over complete graph $G' = (N', E')$ of that shape. E' covers both direct and indirect connections, while E covers only direct connections. Since, for a given shape $E \subseteq E'$ and $N = N'$, we can infer that $G \subseteq G'$, where '$\subseteq$' denotes subgraph relation. Thus part relations of shapes are identified.

The implementation

For being able to detect any, and if necessary, every defined occurrence of the left hand side of a given rule (source shape) within the given initial shape (target shape), a Euclidean transformation of the source shape is to be found identical to a part of the target shape. In our proposal, for each case we have a graph representation of the left hand side of the rule, which we call the source graph, and an over-complete graph representation of the initial shape, where each node has an explicit connection with the nodes that are coincident with the same maximal lines.

Nodes in a given source graph can be categorized as free or non-free. A free node is connected to just one other node whereas a non-free node is connected to more than one node. Free (F) nodes are never at an intersection and are located at the boundaries of the maximal lines. Differently, non-free (NF) nodes are located at the intersection points of two or more non-collinear maximal lines. Therefore, at least one of those edges connected to a NF node is non-collinear with the others. As a result, NF nodes play an important role in the problem of searching for parts.

The detection of parts is handled in two steps. In the first, Euclidean transformations of the selected source subgraph are matched to the target graph in V_{02}. These transformations determine all of the parts of the initial shape, which may match the source shape. In the second, the part relation validation is performed. Considering the source graph attributes, either a full graph matching or a subgraph matching is performed where necessary. If there are free nodes in the source graph a simple graph search is not sufficient to validate part relation. This is due to the fact that when a shape is transformed and

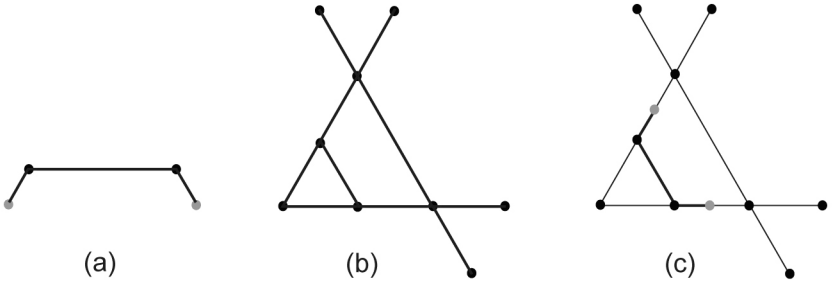


Figure 6
(a) Source graph: free node points are shown in grey color, (b) Target graph, (c) Free nodes are not coincident with any node of the target graph

embedded within an initial shape, the free nodes of the corresponding source graph may or may not be coincident with a node of the corresponding target graph (Figure 6). For this reason, in the second step, the shape algebra extends to include the line segments in $U_{12} \times V_{02}$.

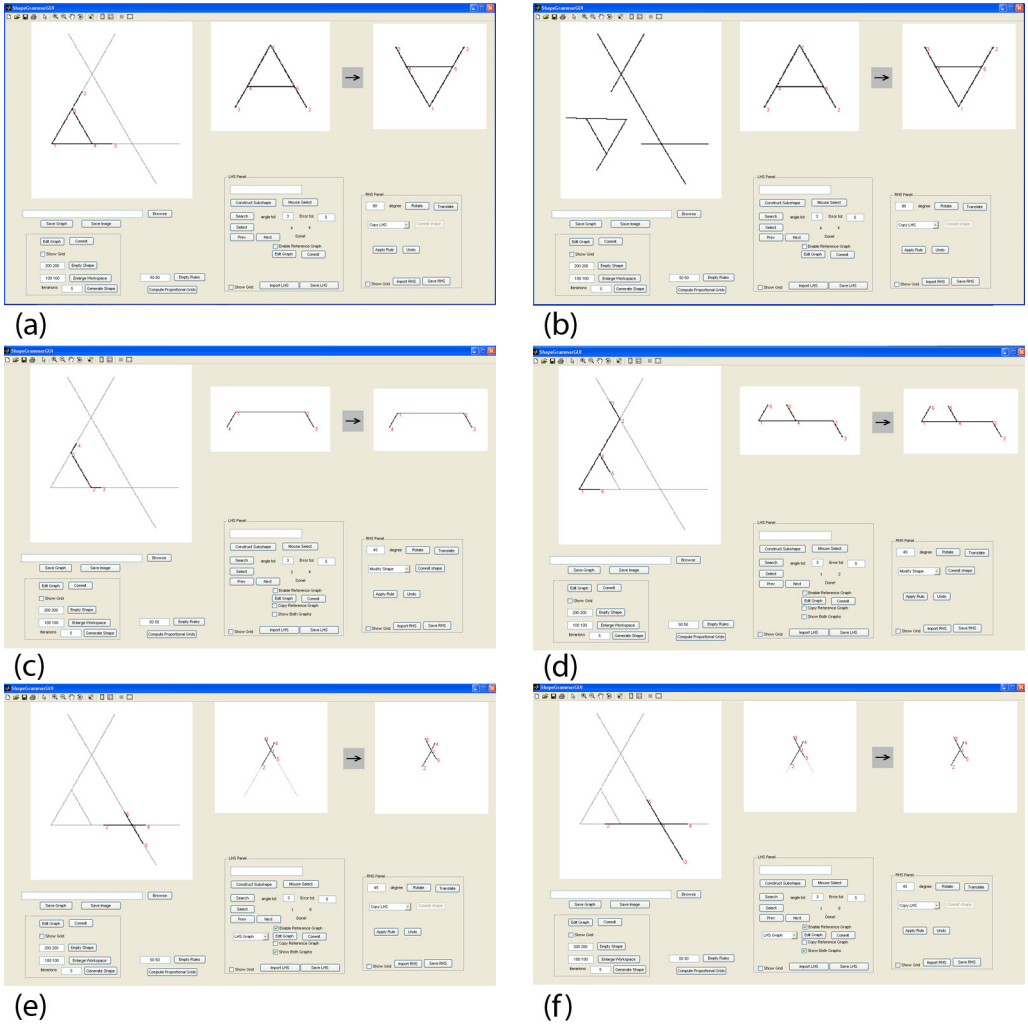
If the shape from the left side of the rule is part of the initial shape, the points corresponding to the free nodes of the source graph introduces new critical points in the target shape in the scope of that particular embedding when the free nodes of the source shape are not coincident with any node of the target graph. If a free source node is not coincident with a target node in the graph, then the point corresponding to the free source node must be coincident with a line in the shape that connects two target nodes, since the source shape is assumed to be a part of the target shape. For embedding such a shape, that point of the initial shape is named a critical point. During shape embedding, the critical point set of the target shape is dynamically extended depending on the position, orientation and scale of the source shape. In this respect, the proposed method dynamically decomposes a given initial shape depending on the left hand side of the rule for each possible mapping.

The proposed part detection method is implemented using MATLAB v7.0 on Windows XP. The user can introduce the initial shape and the rule shape(s) from the graphical user interface. We run the developed program for four exemplary cases in Figure 7: (1) where there are three or more NF nodes in the source graph such that at least three of them are non-collinear, (2) where there are exactly two NF nodes in the source graph, (3) where there are more than two NF nodes in the source graph all of which are collinear (coincident with the same line), and (4) where there are less than two NF nodes in the source graph.

According to the developed part detection algorithm, all possible left hand side rules are in one of the four different cases we define. In the fourth, there are infinitely many transformation possibilities for source shapes. In non-deterministic cases such as this, the technical problem is in the detection of any one of the infinitely many occurrences of the left side within the initial shape. Generating all possible solutions is not feasible both for time and storage limitations of the computing medium. Hence, we introduce a visual communication system that defines constraints for mapping the shape on the left side of the rule to a part of the target shape based on the user's (designer's) intentions.

The communication between the computing machine and the user is done by means of a visual rule definition where the left hand side of the rule is constrained in relation with another shape labeled differently. The left hand side of the rule is defined in a particular way (Figure 7(e,f)). The original left hand side rule is defined in relation with another shape, called *the reference shape*, which is labeled differently (*i.e.* lies in a different layer) during part detection. During the operations, the user defined reference shape is mapped to the V_{02} domain while the original shape (corresponding to the left hand side of the rule) is in $U_{12}.V_{02}$ composite domain. The reference shape is transformed to the V_{02} domain to construct the graph of the reference shape in the usual way. However, the types of the graph nodes are modified to be non-free typed nodes even for the free nodes of the original reference graph. The reference shape helps us determine the possible transformations of the source shape that is used to identify the part of the target shape that matches to the source shape. There is no predefined restriction about the reference shape definition. It may be defined in any way that is meaningful for a user. For the same shape rule displayed in Figure 7(e), changing the scale of the reference shape significantly changes the parts detected in the initial shape (Figure 7(f)).

Figure 7
(a) A sample part detection for Case 1; (b) Rule application is possible once the left hand side is detected in the initial shape as desired by the user; (c) A sample part detection for Case 2 (d) A sample part detection for Case 3; (e) A sample part detection for Case 4; the reference shape is displayed on the same local frame with the left hand side rule within the middle window with lightly colored lines; (f) Second example for part detection for Case 4; the reference shape is different from the one displayed in (e)



Conclusions

Allowing for part detection in non-deterministic cases is the foremost novelty of this implementation. One of the leading motivations in achieving this has been to stay true to the nature of shapes. Therefore, in the presented approach, the elements that represent the shape and parts of it are not abstract or external to the user, but are boundary elements of shapes perceived by the user. The current solution to detect parts that are not predefined is based on the over-complete character of the shape graph representation as well as the user-defined constraints. The decompositions of an initial shape are changeable as

rules are applied, or as constraints are defined, in an interactive implementation where there is no definite description made for the shape

Another aim has been to provide a technical framework that is easy to implement. The computer implementation is consistent only with the algebra defined for shape computation. Constructing the over-complete graph is simply operating on multiple representations in label algebra. The graph data structure provides a temporary representation which (a) facilitates the passage between images in their bitmap form and shapes via attributes; (b) gives us a flexibility in defining perceptual wholes, *i.e.* any open simple traversal, as an alternative to relying on maximal elements; (c) is equipped with algorithms that facilitate search. An extended account of the over complete graph, its connection to algebras, and the related algorithms are discussed by the authors in another paper currently under review for publication. Future aims include extending the computer implementation for three dimensional shape embedding.

Acknowledgements

The research presented is part of a project funded by TÜBİTAK - The Scientific and Technological Research Council of Turkey under the grant number 108E015.

References

Agarwal, M. and Cagan, J.: 1998, A blend of different tastes: the language of coffeemakers, Environment and Planning B: Planning and Design, 25, pp. 205–226.

Chase, S. C.: 1989, Shape and shape grammar - from mathematical model to computer implementation, Environment and Planning B: Planning and Design, 16, pp. 215–242.

Krishnamurti, R.: 1980, The arithmetic of shapes, Environment and Planning B: Planning and Design, 7, pp. 463–484.

Krishnamurti, R.: 1981, The construction of shapes, Environment and Planning B: Planning and Design, 8, pp. 5–40.

Krishnamurti, R. and Giraud, C.: 1986, Towards a shape editor - the implementation of a shape generation system, Environment and Planning B: Planning and Design, 13, pp. 391–404.

Prats, M., Earl, C., Garner, S. and Jowers, I.: 2006, Shape exploration of designs in a style: toward generation of product designs, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 20, pp. 201–215.

Pugliese, M. J. and Cagan, J.: 2002, Capturing a rebel: modeling the Harley-Davidson brand through a motorcycle shape grammar, Research in Engineering Design-Theory Applications and Concurrent Engineering, 13, pp. 139–156.

Stiny, G.: 2006, Shape: Talking about Seeing and Doing, MIT Press, Cambridge, MA.

Stiny, G. and Gips, J.: 1972, Shape grammars and the generative specification of painting and sculpture, in O. R. Petrocelli (ed), The Best Computer Papers of 1971, Auerbach, Philadelphia, pp 125–135.

Tapia, M.: 1999, A visual implementation of a shape grammar system, Environment and Planning B: Planning and Design, 26, pp. 59–74.