Environment and Planning B: Planning and Design 2010, volume 37, pages 664 – 681

doi:10.1068/b36010

Embedding shapes without predefined parts

Hacer Yalim Keles
Computer Engineering Department, Middle East Technical University, Ankara 06531, Turkey;
e-mail: hacerkeles@gmail.com

Mine Özkar ¶
Department of Architecture, Middle East Technical University, Ankara 06531, Turkey;
e-mail: ozkar@metu.edu.tr

Sibel Tari
Computer Engineering Department, Middle East Technical University, Ankara 06531, Turkey;
e-mail: stari@metu.edu.tr
Received 21 January 2009; in revised form 23 August 2009

Abstract. For a practical computer implementation of part embedding in shapes that is also true to their continuous character and the shape grammar formalism, shapes and their boundaries are handled together in composite shape and label algebras. Temporary representations of shapes, termed 'overcomplete graphs', comprise boundary elements of shapes and how they are assembled, and are utilized in a two-phase algorithm that systematically searches for embedded parts. The associated implementation is developed to receive user-defined constraints for an interactive search. In particular, the user-defined reference shape extends the search to nondeterministic cases.

1 Introduction

The theory of shape grammars supports a pragmatic view on shape perception and utilizes visual rules for acting on shapes. A shape is not an entity composed of predefined parts but is temporarily decomposed into its relevant parts according to the desired action. Since rules are visual, there is always the opportunity to perceive the parts, and hence the whole, differently. The recognition of any relevant part within a given shape without a predefined decomposition is a fundamental issue and a technical problem to be resolved for the implementation of visual computing.

There are infinitely many instances of shapes such as that in figure 1, embedded in shapes such as that in figure 2. These instances display Euclidean transformations of rotation, reflection, translation, and scaling. If these figures are to be represented in a digital medium, rather than on paper, the problem is in how to represent both shapes so that all of these instances of the first shape, depending on when we want to use them, are readily available to us to crop off from the second shape.

It is not possible to talk about all of the infinitely many embedded parts of the kind in figure 1 in finite terms. A good portion of these parts will be lost when adhered to an abstract reduction. Yet, it might be feasible to give several constrained sets of these parts, without giving an absolute definition for the initial whole/shape. Elsewhere (Keles et al, 2009) we have introduced a summary of the preliminary form of a practical method that does not compromise the continuous nature of shapes. Pursuing the idea further, here we describe in full detail the technical framework and algorithms developed. The detection of given parts/shapes, such as

Figure 1.

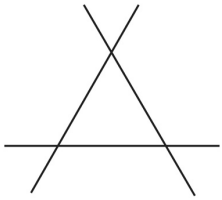

Figure 2.

¶ Corresponding author.

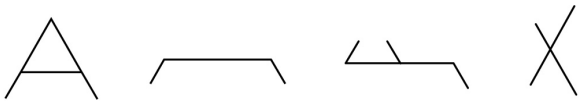

Figure 3.

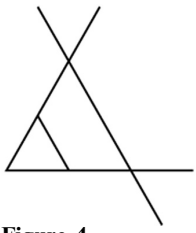

Figure 4.

those in figure 3 in a shape (figure 4), without predefinitions serves to articulate solutions for different technical problems posed with each of these four exemplary cases.

2 Background

The pioneering work of Stiny and Gips (1972), as well as the numerous shape grammars and their implementations that followed it, have brought about a broad research field that is of significant interest to the computational design community. Stiny (2006) recently reinstated the philosophical and technical characteristics of the theory of shape grammars, providing a renewed reference point for this field.

Contributing largely to design research, some of the previous works in the literature used shape grammars to identify and formalize the styles of existing designs and generated new designs with the same style. Others have focused mainly on defining shape rules and shape grammars to generate new design styles. However, many of these shape grammars have not been 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 dwell on specific engineering shape grammars. Heisserman (1994) represents solid shapes as graphs in a generative grammar developed for Queen Anne houses. Some other works (McGill, 2001; Wang and Duarte, 2002) that implement a class of shape grammars referred to as basic shape grammars (Knight, 1999) remain focused on addition rules and their sequences, but not on the technicalities of emergent shape detection.

On the other hand, there are a few works that directly address the implementation of part/subshape detection and of the embedding part relation of shapes. Pointing out that the existing implementations for rule application are limited to certain rule sets in given engineering or design problems and to transforming the entire shape rather than its parts, McCormack and Cagan (2002; 2006) propose a parametric subshape detection method. They decompose shapes into subshapes which are hierarchically ordered on the basis of their constraining relations with other parts. Recognition is then performed by matching the shapes within the same subsets and their combinations. Parametric shape recognition is especially useful for engineering shape grammars with well-defined problems. The hierarchic grouping of subshapes is constructed in accordance with the designer's intention, but is open to the question of how this prejudgment is to constantly change in the design process when new features are meaningful in the determination of new groups for parts of shapes.

A prominent approach proposed for the subshape 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 subshapes (Chase, 1989; Krishnamurti and Giraud, 1986; Tapia, 1999). Tapia's implementation, GEdit, provides a graphical user interface for rules to be defined with orthogonal shapes.

In line with the philosophical perspective regarding the continuous character of shapes and expanding on the guidelines provided by the last two groups of work cited here, this paper illustrates a practically accessible subshape detection method. The proposed method varies technically from the cited work due to its flexible shape representation extending to nondeterministic cases with the help of a reference shape which provides a user-defined context to an otherwise context-free problem. On the technical side, the work introduces a data structure—an overcomplete graph—which, unlike the previous graphs in shape grammar and computer graphics literature (eg Heisserman, 1994), is not fixed.

Furthermore, our approach relies on the user's perception rather than on a nomenclature for the user to learn. While introducing the problem in section 1, we remarked that infinitely many instances of embedded shapes cannot be talked about in finite terms unless one introduces specific constraints. These constraints may take the form of analytic or symbolic abstractions. As exemplified in carriers used by Krishnamurti and Stouffs (2004), fitting a linear form to a maximal line segment and assuming that it continues beyond the given line segment or fitting a higher order function to a curved shape are such analytic abstractions. Instead, we work with a construct that we refer to as a *perceptual whole*, which is not always the maximal element. Our representational units are not abstract or external to the user, but are boundary elements of shapes perceived by the user. When necessary, constraints are introduced by the user according to his or her intentions with the help of a reference shape. We assume nothing beyond what is drawn.

3 The problem of shape representation

The very first steps in embedded part (subshape) detection involve how shapes are represented. Shapes are continuous by nature and 'what you see is what you get'. The ultimate aim is to come up with a representation that does not add or subtract from this continuous nature so that shapes remain as they are. In accordance with the presumption above, shapes, in this paper, are considered with the boundaries of embedded parts as inherent properties of this continuous nature.

Let us look at figure 4 again and propose for the given line shape(s) 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 in V_{ij}, is arbitrary, disputable, and temporary; that is, the U_{12} shape remains as what you see. For figure 4, 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 in Prats et al's (2006) generative study on curved line shapes where they introduce a four-layer construction of contour, decomposition, structure, and design for shape representation. Not implying such constructions, we simply maintain that the intersection points are coincidentally boundary points of some embedded parts. These can be defined as point shapes that are coincident with more than one maximal line in the shape (figure 5).

With this description, boundaries of maximal parts (in this case points) are identified along with those points that are coincident. The shape is enriched to include

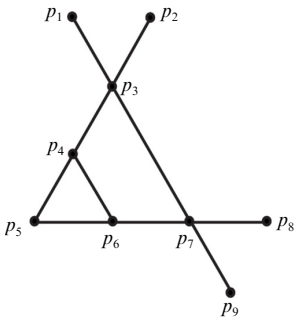

Figure 5. Boundary points: $\{p_1, p_2\, p_4, p_5, p_6, p_8, p_9\}$: intersection points: $\{p_3, p_4, p_5, p_6, p_7\}$; and both boundary and intersection points: $\{p_4, p_5, p_6\}$.

labels for these elements (points) that belong to other algebras (U_{02}). It is thus mapped from the U_{12} algebra into the V_{02} algebra.

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

A maximal line is the Boolean sum of all the infinitely many line segments embedded in it that share either parts or coincident boundaries. Observing the intersection points and the boundary points of the four maximal lines in figure 5, we call these points 'topologically critical points' and identify some line segments that are bounded by these points. These line segments are always bounded by the intersection points, the boundary points of the maximal lines, or both, and are not coincident with any other critical point. They are designated here according to the perceived topology, and serve the purpose of illustrating a generic case. In other cases, they can be assigned, even manually, in many different ways depending on what is intended with the choice of the set of critical points.

Let us define a shape in U_{12} with such segments:

$$S_{U_{12}} = \{l_1, l_2, \ldots, l_n\} ,$$

where l is a line segment. 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 for any shape using the boundary relation for each line segment designated in that shape:

$$\mathrm{M}\colon U_{12} \rightarrow U_{02} ,$$

$$\mathrm{M}(S_{U_{12}}) = \{\mathrm{b}(l_1), \mathrm{b}(l_2), \ldots, \mathrm{b}(l_n)\} ,$$

where b is the boundary operator:

$$\mathrm{b}(l_1) = \{p_{i1}, p_{i2}\} ,$$

and p_{i1} and p_{i2} are the boundary points of the line segment l_i. This operator generates a mapping of the shape from U_{12} to U_{02} where only the topologically critical points are visible. In order to constrain the interpretation of this new point shape, we extend the shift between algebras to the label algebra V_{02} in order to include a set of labels for each point.

$$\mathrm{M}\colon U_{12} \rightarrow V_{02} ,$$

$$\mathrm{M}(S_{U_{12}}) = \{[p_{11}, \mathrm{L}(p_{11})], [p_{12}, \mathrm{L}(p_{12})], [p_{21}, \mathrm{L}(p_{21}), \ldots, [p_{n1}, \mathrm{L}(p_{n1})], [p_{n2}, \mathrm{L}(p_{n2})]\} ,$$

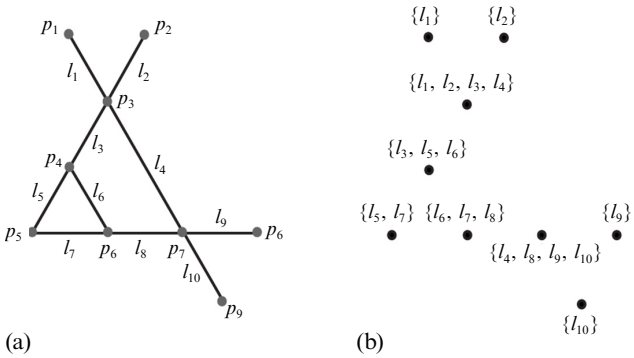

Figure 6. (a) Shape in $U_{12}V_{02}$ with labels for line segments and points shown for illustrative purpose; (b) shape in V_{02}.

where 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 (figure 6).

The shape obtained after this mapping contains a finite number of points and associated labels. Labels encode the information of how a point connects to other points. Labels can be changed, altering the connection information for alternative readings of the shape in V_{02} without altering the shape. We can also represent a shape using different V_{02} representations as dictated by the selection of critical points. We can add all these V_{02} representations to get yet another V_{02} representation, all without altering the shape.

3.2 Attributed representation

For a computer implementation we define the points and their relations in V_{02} as an attributed undirected graph, a temporary data structure to facilitate their representation. The nodes (vertices) of the graph refer to the points in V_{02}. The nodal attributes refer to labels which code the connectivity between nodes. Thus, the graph edges are indirectly implied by the nodal attributes. In this paper we will refer to these attributed undirected graphs as 'graph representations of shapes'.

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:

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

where L(n) is the set of labels associated with the point corresponding to node n. There are two kinds of attributes stored for each node in the simple undirected graph representation:

1. *Node position*: local image space coordinates of the points of the shape in V_{02}.
2. *Edge information*: edges code in V_{02} how the elements in U_{02} bound parts of the U_{12} shape. In other words, edges very abstractly represent the elements (of choice induced by the selection of critical points) that are parts of the U_{12} shape and the boundaries of which coincide with the U_{02} shape.

For linear shapes the connection between two nodes is the unique line that passes through the node coordinates. Hence, it is sufficient to keep a simple flag that represents a linear connection. Merely for efficiency considerations for the algorithms we present in the upcoming sections, in the linear case, we store node-to-node connection angles which could have been calculated from the node position information (figure 7).

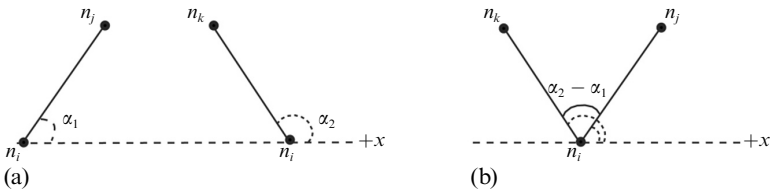

Figure 7. (a) Sample node-to-node connection angles (that is, the positive angle with the positive x-direction for the edge connecting the nodes n_i to n_j and n_k): α_1 and α_2. (b) The edge connection angles between the edges connecting (n_i, n_k) and (n_i, n_j): $(\alpha_2 - \alpha_1)$.

The mapping of a U_{12} shape to a V_{02} shape as depicted in figure 6 for linear shapes readily applies to curved shapes when the line segments are replaced with arbitrary segments induced by the U_{02} shape. Linear or not, segments of U_{12} elements, have a boundary elements set in V_{02} (figure 8). Identical attributes, which are as specific as angles in the linear case, indicate collinearity, continuity, and hence a perceptual whole, in this case a maximal line. More generally, we may perceive a U_{12} element, say $(s_5 \cup s_3 \cup s_2)$ in figure 8(a), as a perceptual whole whose boundaries are given by the points p_5 and p_2. If two edges connecting through a node have similar attributes, we can choose not to perceive the node as a critical point leading to an alternative traversal of the graph. This implies an alternative perception in U_{02} and V_{02} without altering the U_{12} shape. Generically, each segment can be described in a canonical frame (figure 9) to facilitate comparison without fitting a curve and respecting the segment as it is. The nodal position of n_i is mapped to $(0, 0)$ and the nodal position of n_j is mapped to $(1, 1)$ in the canonical representation of a curve segment bounded by n_i and n_j. The crucial point here is to make the node order compatible for the node pairs of both graphs during the matching of two graph line segments.

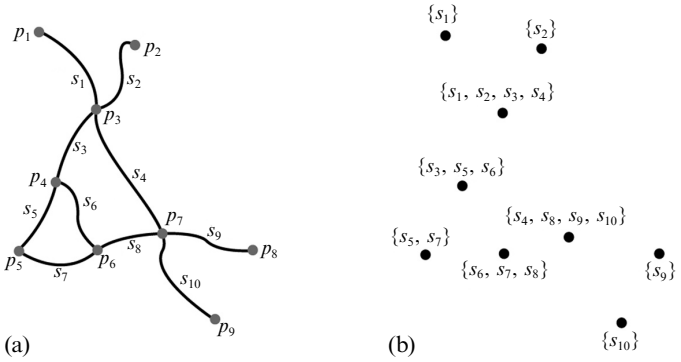

Figure 8. Linear or not, segments of U_{12} elements have boundary elements set in V_{02}.

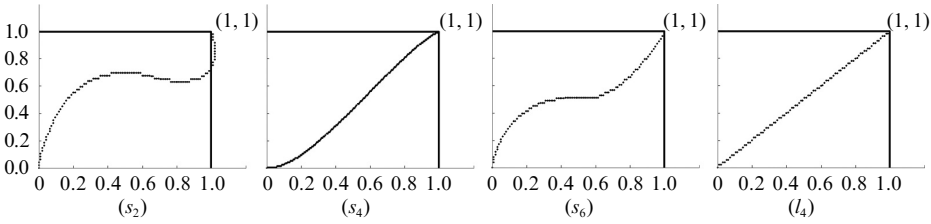

Figure 9. Sample canonical representations for the curved line segments s_2, s_4, and s_6 of the shape in figure 8 and the straight-line segment l_4 in figure 6. For straight-line segments, the canonical representation is the same (except for the density of the plotted points).

Once the segments have been represented in a canonical frame with all the position and scale effects filtered out, even a simple vectorial distance can be used to measure the differences between two segments. Notice the resemblance between the canonical frame representations of s_4 and l_4. In general, matching is a separate issue. One can even use elastic curve difference measurement techniques such as dynamic time warping (Sakoe and Chiba, 1978) on segments without normalizing their scales.

4 Handling identity relations of shapes

A shape can be perceived in different instances where its scale, orientation, and position change. All these instances are identical shapes under Euclidean transformations. In this section we seek correlation between the graph representations of shapes in order to detect identity-embedding relations between shapes (figure 10).

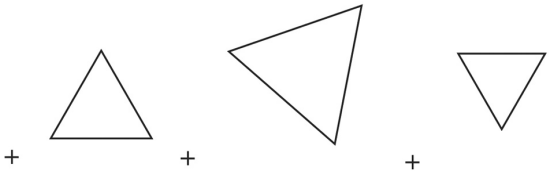

Figure 10. Three different instances that are shape equivalent under Euclidean transformations.

The mapping function described in section 3 is not a one-to-one relation where each shape is uniquely represented with a graph and 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 undergo Euclidean transformations. Only the positions of the points (nodes) change. To seek 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. This is a bijection between the node sets of two exemplary graphs G_1 and G_2,

$$\mathrm{f}\colon \mathrm{N}(G_1) \rightarrow \mathrm{N}(G_2)\ ,$$

such that any two nodes n_i and n_j of G_1 are adjacent in G_1 if and only if $\mathrm{f}(n_i)$ and $\mathrm{f}(n_j)$ are adjacent in G_2. Moreover, for any two nodes n_j and n_k of G_1 where the node n_i has an adjacency relation:

(1) The edge connection angle between $\langle n_j, n_i, n_k \rangle$ nodes of G_1 centered at n_i and the corresponding angle centered at $\mathrm{f}(n_i)$ in G_2, $\langle \mathrm{f}(n_j), \mathrm{f}(n_i), \mathrm{f}(n_k) \rangle$, should be equivalent.

(2) The edge length ratio $r_{G_1} = \|\mathrm{P}(n_j) - \mathrm{P}(n_i)\| / \|\mathrm{P}(n_k) - \mathrm{P}(n_i)\|$ where $(0 < r_{G_1} \leqslant 1)$ of G_1 and $r_{G_2} = \|\mathrm{P}[\mathrm{f}(n_j)] - \mathrm{P}[\mathrm{f}(n_i)]\| / \|\mathrm{P}[\mathrm{f}(n_k)] - \mathrm{P}[\mathrm{f}(n_i)]\|$ (where $0 < r_{G_2} \leqslant 1$) of G_2 should be equivalent. Note that $\mathrm{P}(n)$ gives the local coordinates of a node n.

The shape identity relation partitions the set of graphs into equivalence classes. These classes are subsets of the structural isomorphism classes of graphs. In other words, being in the same isomorphism class for two graphs is a necessary but insufficient condition for being in the same shape equivalence class. The two graphs should also satisfy the constraints defined in (1) and (2).

5 Handling part relations of shapes

Let us assume that we are to detect shape (a) as a part of shape (b) in figure 11. There is more than one occurrence of (a) within (b) at different scales and orientations. When the simple graph of (b) is traversed to obtain the subshapes which are shape equivalent

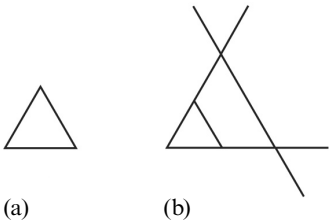

Figure 11. Sample shape (a) to be detected in shape (b).

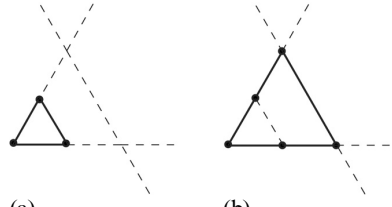

Figure 12. Subgraphs for the graph of shape in figure 11(b).

to (a), we obtain two simple subgraphs for which nodes and node connections are shown in figure 12. Although both are shape equivalent to figure 11(a) under some Euclidean transformations, figure 12(b) is not graph equivalent to figure 11(a) as it has more nodes than the boundaries of its maximal lines. Instead, the corresponding subgraph representation of figure 11(b) should be as in figure 13 where the only nodes that define the subgraph are topologically critical points.

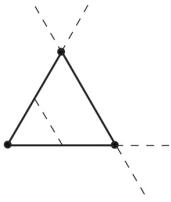

Figure 13. A subgraph of the graph of the shape in figure 11(b).

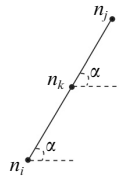

Figure 14. Node connection angles between node pairs (n_i, n_k) and (n_k, n_j).

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 overcomplete 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 12(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 here onwards call an 'overcomplete graph representation of a shape', all implicit connections are made explicit, and the subgraph shown in figure 13 becomes a valid subgraph of the graph in figure 11(b).

Let us call a graph $G' = (N', E')$ an overcomplete 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 14). Then n_i is connected to both n_k and n_j. 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 overcomplete 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 overcomplete 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 the subgraph relation.

6 The two-step detection of parts in an initial shape

In a nondeterministic case, for the left-hand side of the visual rule in figure 15 to be embedded in the initial shape in figure 16 as part of a computation, the technical problem is the detection of any one of the infinitely many occurrences of the left-hand side within the initial shape.

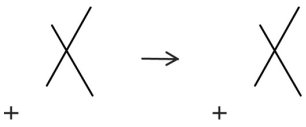

Figure 15.

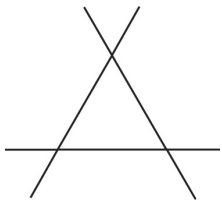

Figure 16.

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. Assume that $G_S = (N_S, E_S)$, as the graph representation of the left-hand side of the rule, is the source graph and $G_T = (N_T, E_T)$, as the overcomplete 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 nonfree. A free node is connected to just one other node whereas a nonfree node is connected to more than one node. Free nodes are never at an intersection and are located at the boundaries of the maximal lines. Differently, nonfree nodes are located at the intersection points of two or more noncollinear maximal lines. Therefore, at least one of those edges connected to a nonfree node is noncollinear with the others. As a result, nonfree 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 because, when a shape is transformed and 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 17). For this reason, in the second step, the shape algebra extends to include the line segments in $U_{12} V_{02}$.

If the shape from the left-hand side of the rule is part of the initial shape, a free node of the source graph is either coincident with a node of the target graph or coincident with a boundary point of a subpart embedded in a line connecting two target nodes. In the latter case, the free node introduces a new critical point in the target shape, in the scope of that particular embedding.

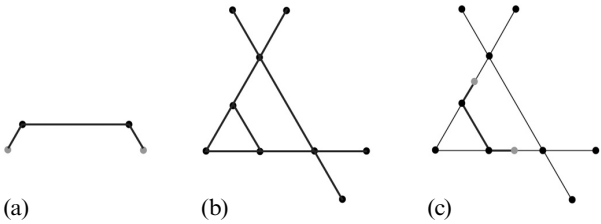

Figure 17. (a) Source graph: free node points are shown in grey; (b) target graph; (c) free nodes are not coincident with any node of the target graph.

If a free source node is not coincident with a target node in V_{02}, then the point corresponding to the free source node must be coincident with a line in U_{12} 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 in $U_{12}V_{02}$ algebra, 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.

7 The algorithm and four exemplary cases

The algorithmic details of the two steps are explained by referring to the four exemplary cases briefly illustrated in figure 3. The initial shape selected for the exemplary cases has already been given in figure 4.

7.1 First pass

(1) Construct the source graph $G_S = (N_S, E_S)$ and overcomplete target graph $G_T = (N_T, E_T)$.
(2) Enumerate the nodes in the target graph $(n_1^T, n_2^T, \ldots, n_k^T)$, where k is the total number of nodes in the overcomplete target graph.
(3) From the source graph, select a subset of the source nonfree (NF) nodes in order to construct a candidate transformation that transforms the source graph nodes into the target graph domain to detect parts. There are four different cases to select this subset according to the number of nonfree nodes in the source graph.

Case 1: *There are three or more nonfree nodes in the source graph such that at least three of them are noncollinear* [figure 3(a) and figure 18]:
1. Select three nonfree source nodes $S_{NF3} = \{n_1^S, n_2^S, n_3^S\}$ and construct the corresponding three-node vertex-induced subgraph of the source graph $G_S[S_{NF3}] = (N_{S3}, E_{S3})$.
2. Create a set of candidate node mappings π by generating all three-node permutations of the target nodes. There are $k(k-1)(k-2)$ permutations. Four sample mappings are shown in the first row of figure 18.
3. Construct a three-node vertex-induced subgraph of the target graph for each permutation $T_{N3} = \{n_i^T, n_j^T, n_m^T\}$ as $G_T[T_{N3}] = (N_{T3}, E_{T3})$, where $1 \leqslant i, j, m \leqslant k$.
4. For each node mapping, $\mu(n_1^S, n_2^S, n_3^S) = (n_i^T, n_j^T, n_m^T)$, $(\mu \in \pi)$:
(a) Compare the subgraph relation between $G_S[S_{NF3}]$ and $G_T[T_{N3}]$. $G_S[S_{NF3}]$ must be a subgraph of $G_T[T_{N3}]$ according to the particular node mapping μ.
(b) Node distance ratios among the source nodes in N_{S3} must be compatible with the node distance ratios among the mapped target nodes in N_{T3}.
(c) Edge connection angles as illustrated in figure 7 among the source nodes in N_{S3} must be compatible with the node connection angles among the mapped target nodes in N_{T3}.
(d) List every mapping which satisfies these three conditions.

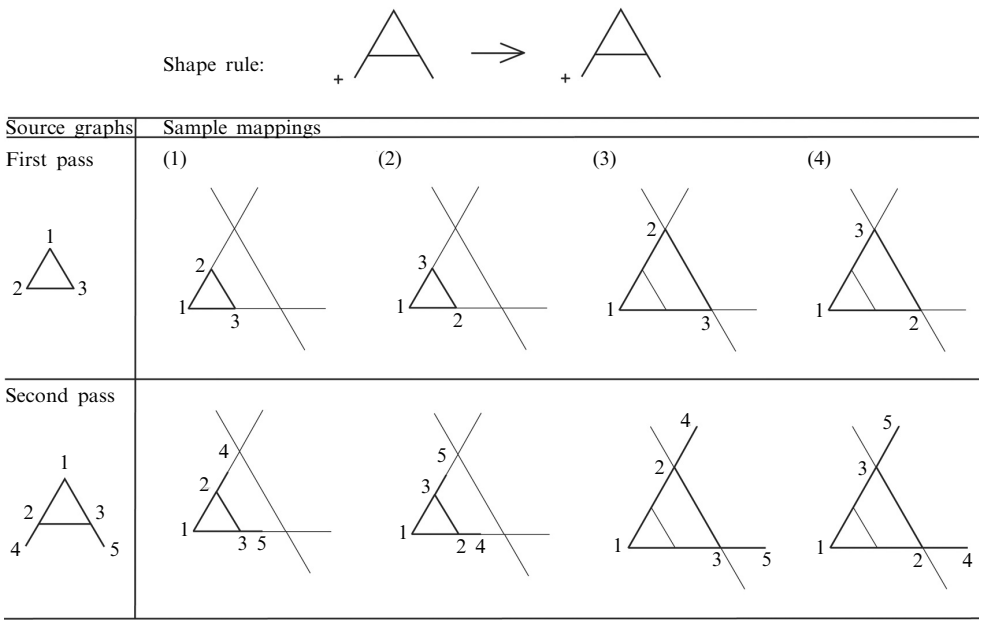

Figure 18. Example rule for case 1 is displayed for four sample mappings (1) – (4). The candidate mappings are displayed in the first row. A three-node source subgraph is selected from the source graph and shown in the left-hand column of the first row. In the second pass illustrated in the second row, the source graph is matched according to the transformation computed in the first pass. Transformations in the second pass can be followed easily by tracing the node correspondences between the graphs used in the first and second passes with the indexes.

5. For each compatible mapping in the list, compute the transformation matrix $\mathbf{\Gamma}$. Since the source node coordinates and the corresponding target node coordinates are known for three nonlinear points, the transformation matrix can be computed uniquely.

Case 2: *There are exactly two nonfree nodes in the source graph* [figure 3(b) and figure 19]:
1. Select two nonfree source nodes $S_{\mathrm{NF2}} = \{n_1^{\mathrm{S}}, n_2^{\mathrm{S}}\}$ and construct the corresponding two-node vertex-induced subgraph of the source graph $G_{\mathrm{S}}[S_{\mathrm{NF2}}] = (N_{\mathrm{S2}}, E_{\mathrm{S2}})$. Selected nonfree source graph nodes are depicted with the indexes 1 and 2 in the left-hand column of the first row in figure 19.
2. Create a set of candidate mappings π by computing all two-node permutations of the target nodes. Four sample mappings are displayed in the right-hand column of the first row in figure 19.
3. Construct a two-node vertex-induced subgraph of the target graph for each permutation $T_{N2} = \{n_i^{\mathrm{T}}, n_j^{\mathrm{T}}\}$ as $G_{\mathrm{T}}[T_{N2}] = (N_{\mathrm{T2}}, E_{\mathrm{T2}})$, where $1 \leqslant i, j \leqslant k$.
4. There are two-node mappings between the source graph and the target graph. Therefore, a unique transformation matrix cannot be computed. Two artificial reference points can be introduced to compute the transformation matrices uniquely for both source and target node pairs. Assume that an external line passes through one of the two nonfree source nodes, say n_2^{S}, perpendicular to the line connecting two nonfree source nodes. The length of the external line is equal to the line connecting two nonfree source nodes and the selected nonfree node is coincident with the external line at the line midpoint. The boundary points of this newly created external line are the external reference points for the source nodes: $\{r_1^{\mathrm{S}}, r_2^{\mathrm{S}}\}$ as shown in the source graph of the first row in figure 19. Similarly, an external line is drawn which passes through the target node corresponding to the selected nonfree source node, say n_j^{T},

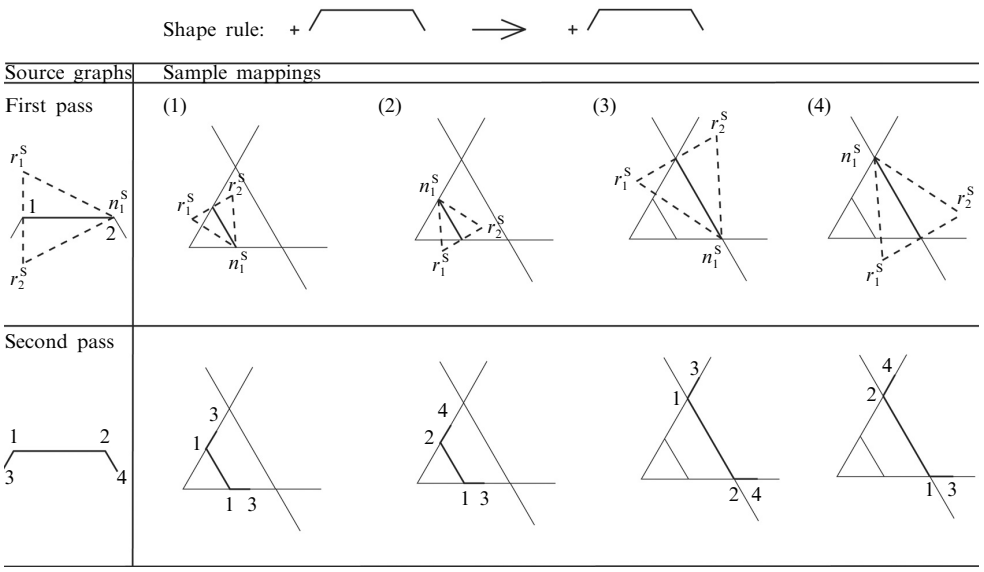

Figure 19. Example rule for case 2 is displayed for four sample mappings in (1) – (4). The graph displayed with dashed lines in the first row contains nodes that are constructed with external reference points explained in case 2(4).

according to the particular mapping. The line length is equal to the length of the line which connects the two target nodes and the selected target node coincides with the external line at the line midpoint. The boundary points of this line are the reference points for the target node pairs $\{r_1^{\mathrm{T}}, r_2^{\mathrm{T}}\}$.

5. There are two possible transformations that map the source graph nodes to the target subgraph nodes. One of the transformations is a reflection of the other about the axis that passes through the line connecting the two source/target nodes. Construct these mappings as $\mu_1(n_1^{\mathrm{S}}, r_1^{\mathrm{S}}, r_2^{\mathrm{S}}) = (n_i^{\mathrm{T}}, r_1^{\mathrm{T}}, r_2^{\mathrm{T}})$, and $\mu_2(n_1^{\mathrm{S}}, r_1^{\mathrm{S}}, r_2^{\mathrm{S}}) = (n_i^{\mathrm{T}}, r_2^{\mathrm{T}}, r_1^{\mathrm{T}})$.

6. For each mapping, μ_1 and μ_2, compute a unique transformation matrix $\mathbf{\Gamma}$.

Case 3: *There are more than two nonfree nodes in the source graph all of which are collinear (coincident with the same line)* [figure 3(c) and figure 20]:

1. Select two nonfree source nodes randomly and compute the transformation matrices following the algorithmic steps described in case 2.

Case 4: *There are less than two nonfree nodes in the source graph* [figure 3(d) and figure 21]:

1. If a reference shape is provided (see the discussion related to the reference shapes below in this section):

(a) Compute the reference shape graph $G_{\mathrm{R}} = (N_{\mathrm{R}}, E_{\mathrm{R}})$.

(b) Compute the transformation matrix as in case 1 or case 2, depending on the number of reference nodes. Note that all the reference graph nodes are of the nonfree type by definition of the reference graphs.

2. If there is no reference shape, assume that all nodes of the source shape are of the nonfree type. Compute the transformation matrices as in case 1 or case 2, depending on the number of source nodes.

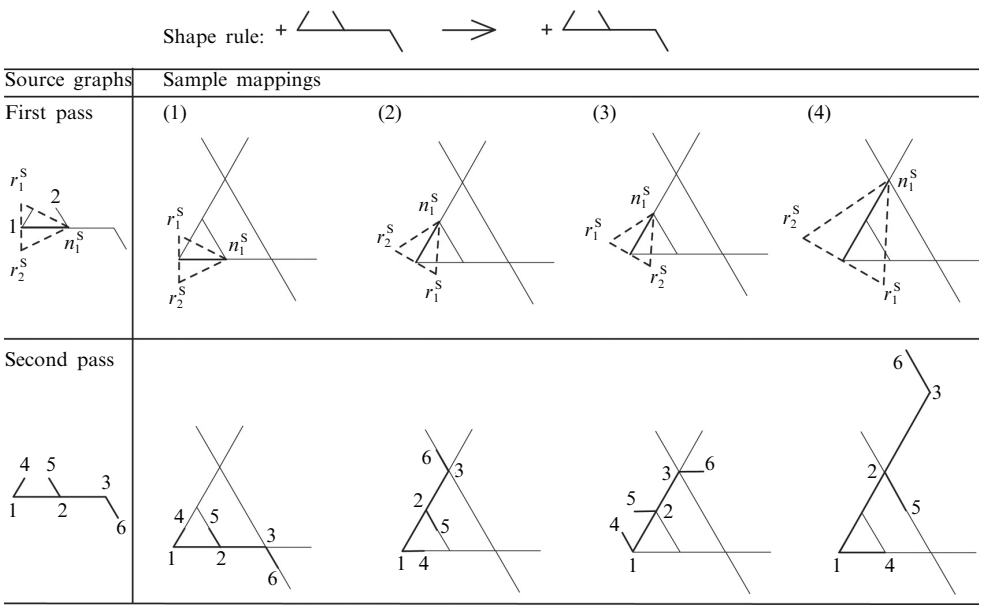

Figure 20. Example rule for case 3 is displayed for four sample mappings in (1) – (4). The mappings in (1) and (2) are acceptable mappings for which source shape is part of the target shape, while (3) and (4) are rejected mappings in the second pass of the algorithm.

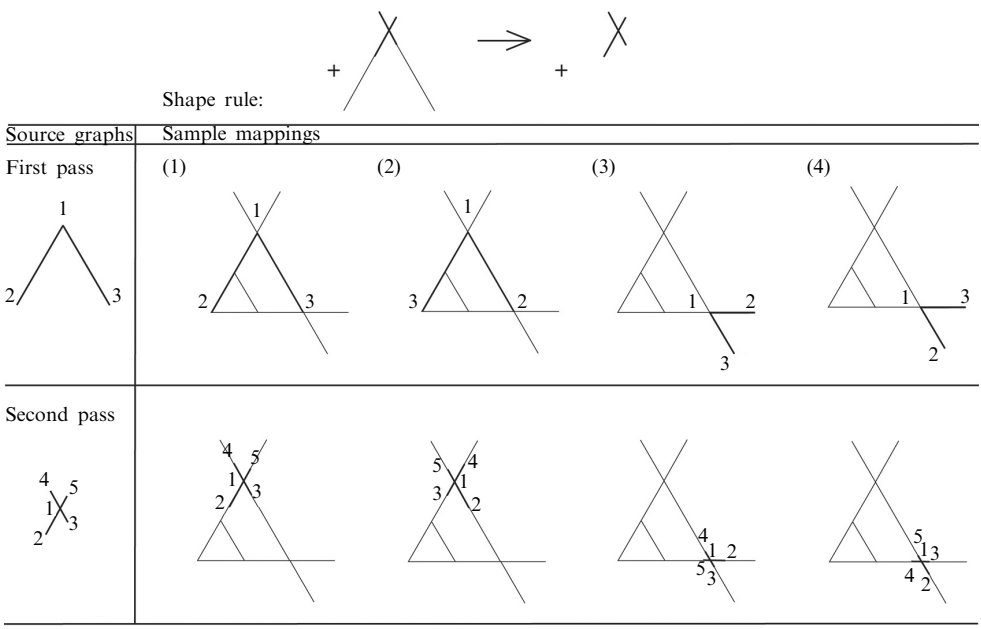

Figure 21. Example rule for case 4 is displayed for four sample mappings in (1) – (4). A user-defined reference shape is introduced along with the left-hand side of the rule, displayed with light-weighted lines in the rule definition. Since all the nodes in a reference shape graph are of the nonfree type, each critical point of the reference shape is coincident with a critical point of the target shape during the computation of candidate mappings in the first pass.

7.2 Second pass

(1) For each possible mapping computed in the first step, find node associations among the source nodes and the target nodes as:

(a) Transform all the nodes (free and nonfree nodes) of the source graph using the transformation matrix ($\mathbf{\Gamma}$) computed in the first step.

(b) For all the transformed nonfree source nodes, compute their correspondences, μ, with the target nodes by comparing coordinates. If there is any nonfree source node which does not correspond to any target node, discard this mapping from the candidate mapping list; for example, in figure 20, the nonfree indexed as 3 in the sample mapping (4).

(c) Compute the node correspondence, μ, for all free source nodes:

i. If the transformed free source node is coincident with a node of the target graph, associate the target node index with the corresponding free source node; for example, in figure 20, the free node indexed as 4 in the sample mapping (4).

ii. If the transformed free source node is not coincident with a node of the target graph, but coincident with a line segment that connects two target nodes, associate the free source node with the set of target nodes which are coincident with the boundary points of the corresponding line segment; for example, in figure 21, all free nodes coincide with a line for the displayed sample mappings.

iii. If the transformed free source node is not coincident with a target node or with a line segment that connects two target nodes, discard this mapping from the candidate mapping list; for example, in figure 20, nodes indexed as 4, 5, and 6 in sample mapping (3).

(2) Control the node connection relations to validate each mapping, μ:

(a) If each source node is associated with just one target node, then check for each $(n_i^{\mathrm{S}}, n_j^{\mathrm{S}}) \in E_{\mathrm{S}}$, whether there exists a relation $[\mu(n_i^{\mathrm{S}}), \mu(n_j^{\mathrm{S}})] \in E_{\mathrm{T}}$. If there exists any $(n_i^{\mathrm{S}}, n_j^{\mathrm{S}}) \in E_{\mathrm{S}}$ such that $[\mu(n_i^{\mathrm{S}}), \mu(n_j^{\mathrm{S}})] \notin E_{\mathrm{T}}$, this mapping is deleted from the candidate mapping list.

(b) If, for some of the free source nodes, there is more than one associated target node [see (1)(c)ii], assume that $(n_i^{\mathrm{S}}, n_j^{\mathrm{S}}) \in E_{\mathrm{S}}$ and $\mu(n_j^{\mathrm{S}}) = \{n_a^{\mathrm{T}}, n_b^{\mathrm{T}}\}$.

i. If $n_a^{\mathrm{T}} \neq \mu(n_i^{\mathrm{S}})$ and $n_b^{\mathrm{T}} \neq \mu(n_i^{\mathrm{S}})$, there must be $[\mu(n_i^{\mathrm{S}}), n_a^{\mathrm{T}}] \in E_{\mathrm{T}}$ and $[\mu(n_i^{\mathrm{S}}), n_b^{\mathrm{T}}] \in E_{\mathrm{T}}$ with the same connection angles. Otherwise, discard this mapping.

ii. If $n_a^{\mathrm{T}} = \mu(n_i^{\mathrm{S}})$ and $n_b^{\mathrm{T}} \neq \mu(n_i^{\mathrm{S}})$, then there must be $[\mu(n_i^{\mathrm{S}}), n_b^{\mathrm{T}}] \in E_{\mathrm{T}}$. Otherwise, discard this mapping.

iii. If $n_b^{\mathrm{T}} = \mu(n_i^{\mathrm{S}})$ and $n_a^{\mathrm{T}} \neq \mu(n_i^{\mathrm{S}})$, then there must be $[\mu(n_i^{\mathrm{S}}), n_a^{\mathrm{T}}] \in E_{\mathrm{T}}$. Otherwise, discard this mapping.

(c) For each valid candidate mapping remaining in the list, the source shape is accepted as part of the target shape with respect to the corresponding transformation.

According to the developed part detection algorithm, all possible left-hand sides of the rules are in one of the four different cases defined in the first step of the algorithm (see step 3). The last case (case 4) covers the rules in which the source graph contains less than two nonfree nodes. There are infinitely many transformation possibilities for this class of source shapes. 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-hand side of the rule to a part of the target shape based on the user's (designer's) intentions.

In this system the communication between the computing machine and the user is similarly done by means of visual rule definition in U_{12}. However, the left-hand side of the rule is defined in a particular way (figure 21). The original left-hand side of the rule is defined in relation with another shape, called the *reference shape*, which is

labeled differently (that is, 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 the $U_{12}V_{02}$ 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 graph nodes are modified to be nonfree-type nodes even for the free nodes of the original reference graph. The reference shape determines 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 example, in figure 22, a change in the scale of the reference shape is introduced without changing the visual rule applied in figure 21. Such a change in the reference shape significantly changes the parts detected in the initial shape. The solution proposed for this type source of graph searches enables the user to embed any source shape to a target shape by characterizing a reference shape (figure 23). In the case that no reference shape is provided to the system, the mapping constraints are defined by the system by redefining the node types of the source graph: for example, the set of mappings can be constrained by redefining all the free nodes as nonfree. This can be changed according to the user's intentions.

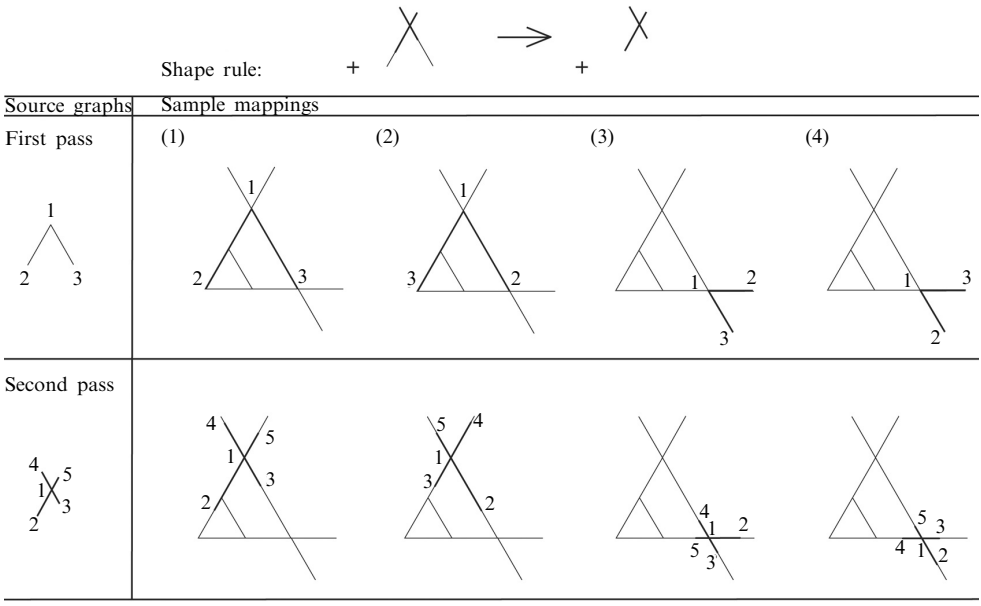

Figure 22. The application of the same rule in figure 21 with a different reference shape.

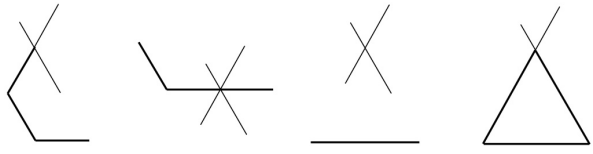

Figure 23. Sample reference shapes for the same rule in figure 21. Reference shapes are emphasized with bold lines.

8 Further discussion

There may be cases in which the maximal lines do not intersect or in which their virtual extensions intersect (figure 24). In our framework, such cases are no different than case 4. In figures 25 and 26 we show two possible rules with varying reference shapes for the case with two parallel lines. In the first rule given in figure 25, neither the shape nor the added reference shape contains any intersection points. Yet, the computer program is able to detect the embedded part (see the online appendix, figure A8, http://dx.doi.org.10.1068/b26010ap). Two possible embeddings obtained with the second choice of the reference shape as displayed in figure 26 are shown in appendix figures A9 and A10.

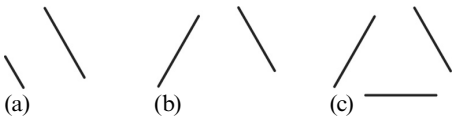

Figure 24.

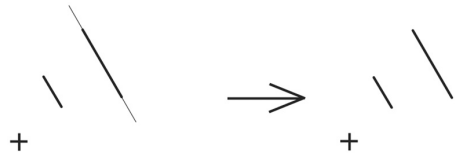

Figure 25.

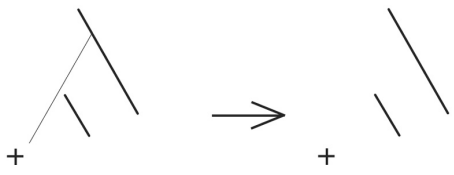

Figure 26.

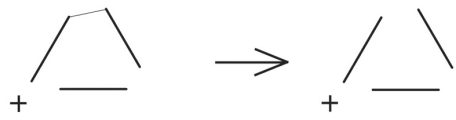

Figure 27.

The case shown in figure 24(c) requires a separate discussion. Whereas the embedding of shapes in figure 24(a) and 24(b) are indeterminate, the embedding of shape shown in figure 24(c) is determinate. Here, the reference shape can help in constraining the problem further as shown in figure 27.

At this point, it is worth emphasizing once again that all that the method requires is critical points that serve merely as registration marks. These registration marks are to be consistent only in the scope of a particular embedding. They carry the same meaning in both the source and the target shapes. In a case where perceptual wholes do not intersect, another criterion may be sought. In this paper we emphasize the importance of the user-defined reference shape to provide registration marks. Even though it is appealing to complete the shape into a triangle by extending the three maximal line segments beyond their drawn scope and marking the three corners of the virtual triangle as critical points, we avoid imposing such interpretations on the shape.

Since we accept that shape is continuous in nature, we consider its parts to be topological neighborhoods on it. Some that are more distinct than others are the critical neighborhoods. As a critical neighborhood shrinks into a point, it serves as a registration mark. In the absence of any such point, all points on the U_{12} shape are equally critical and the embedding problem reduces to a partial and deformable template matching. Such an approach has many practical difficulties. As a part of our future work, we are developing weighted and labeled registration marks to solve the above-mentioned problem efficiently while avoiding any reduction of a particular shape to an analytic form.

The extension of our framework to solid shapes is not straightforward, but possible. Nodes of the overcomplete graph store the perceived topology in terms of how the parts are assembled. Thus these are linkage elements. In the case of U_{12} the linkage between parts is naturally coded by points which are nothing but the boundaries of

one dimensional manifolds in a two-dimensional space. In the case of U_{33}, the nodes should store two-dimensional manifolds in a three-dimensional space. Labels store how the parts are combined via linkage elements and what the parts are so as to construct graph edges.

9 Conclusion

This paper gives a detailed explanation of the technical framework developed for a working computer implementation to handle the embedding relations of shapes (figures A1 – A10 in appendix A). The proposed system is an alternative to previous approaches to the problem. It explores graph data structure to temporarily represent boundary elements of shapes as well as how they are assembled. With the associated algorithms, this structure enables a systematic search for parts.

Our solution to detect parts that are not predefined is based on the overcomplete character of the shape graph representation as well as on 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.

One of our aims has been to stay true to the nature of shapes. Therefore, 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. As seen in the earliest shape computation descriptions by Stiny and Gips (1972), boundaries of shapes are elements of visual computing. The generative rules that Stiny and Gips then specified in their grammar for a series of oil paintings where weighted planes overlap to give way to new shapes, are constituted of line shapes that define areas to be shaded in later on. In visual computations, boundaries of maximal shapes often serve as references to look for or identify parts: for example, drawing an outline first to later shade in the plane. More importantly, boundaries are also treated as shapes. Thus we can say that one of the motivations for the idea of utilizing multiple shape and label algebras for embedding parts is coming from design.

Another one of our aims has been to provide a technical framework that is easy to implement and that benefits from shape grammar formalism. The overcomplete graph is constructed simply by operating on multiple representations in label algebra. To recapitulate, the graph data structure is a temporary representation. It facilitates the passage between images in their bitmap form and shapes via attributes and is equipped with algorithms to facilitate search. It also brings flexibility in defining many kinds of perceptual wholes, rather than relying on maximal elements.

Although graph representations are reductions of continuous shapes to discrete nodes and their labeled connections, the approach presented here utilizes shape algebras and part relations true to visual thinking. Not relying on any geometric notion or symbolic representation, it works with perceptual wholes and no primary features unless specified by the user. In this paper, topologically critical points are always either the boundaries of perceptual wholes or the boundaries of segments embedded in these perceptual wholes which have coincidence, commonly known as intersection points. These boundaries induce what the parts are and their labels induce how they correlate.

Acknowledgements. The research presented is part of a project funded by TÜBİTAK — The Scientific and Technological Research Council of Turkey — under grant number 108E015. We are indebted to the anonymous reviewers for their constructive comments and suggestions.

References

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

Chase S C, 1989, "Shapes and shape grammars: from mathematical model to computer implementation" *Environment and Planning B: Planning and Design* **16** 215 – 242

Heisserman J, 1994, "Generative geometric design" *IEEE Computer Graphics and Applications* **14**(2) 37 – 45

Keles H Y, Özkar M, Tari S, 2009, "Revisiting shape embedding", in *Proceedings of eCAADe, European Computer Aided Architectural Design Education — Computation: The New Realm of Architectural Design* Eds G Cagdas, B Colakoglu (Istanbul Technical University and Yıldiz Technical University, Istanbul) pp 229 – 236

Knight T W, 1999, "Shape grammars: six types" *Environment and Planning B: Planning and Design* **26** 15 – 31

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

Krishnamurti R, 1981, "The construction of shapes" *Environment and Planning B* **8** 5 – 40

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

Krishnamurti R, Stouffs R, 2004, "The boundary of a shape and its classification" *Journal of Design Research* **4**(1), doi: 10.1504/JDR.2004.009843

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, 2006, "Curve-based shape matching: supporting designers' hierarchies through parametric shape recognition of arbitrary geometry" *Environment and Planning B: Planning and Design* **33** 523 – 540

McGill M C, 2001, "A visual approach for exploring computational design", Master of Science in Architecture Studies thesis, School of Architecture and Planning, Massachusetts Institute of Technology, Cambridge, MA

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

Pugliese M J, 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** 139 – 156

Sakoe H, Chiba S, 1978, "Dynamic programming algorithm optimization for spoken word recognition" *IEEE Transactions on Acoustics, Speech and Signal Processing* **26**(1) 43 – 49

Stiny G, 2006 *Shape: Talking about Seeing and Doing* (MIT Press, Cambridge, MA)

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

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

Wang Y F, Duarte J P, 2002, "Automatic generation and fabrication of designs" *Automation in Construction* **11** 291 – 302