PART EMBEDDING FOR SHAPE GRAMMARS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HACER YALIM KELEŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

JULY 2010

Approval of the thesis:

PART EMBEDDING FOR SHAPE GRAMMARS

submitted by **HACER YALIM KELEŞ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering**

Prof. Dr. Sibel Tarı
Supervisor, **Computer Engineering, METU**

Assoc. Prof. Dr. Mine Özkâr
Co-supervisor, **Architecture, METU**

Examining Committee Members:

Prof. Dr. M. Ruşen Geçit
Engineering Sciences, METU

Prof. Dr. Sibel Tarı
Computer Engineering, METU

Assoc. Prof. Dr. Halit Oğuztüzün
Computer Engineering, METU

Assist. Prof. Dr. Tolga Can
Computer Engineering, METU

Assoc. Prof. Dr. Emre Özgen
Psychology, Bilkent University

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: HACER YALIM KELEŞ

Signature :

ABSTRACT

PART EMBEDDING FOR SHAPE GRAMMARS

Yalım Keleş, Hacer
Ph.D., Department of Computer Engineering
Supervisor : Prof. Dr. Sibel Tarı
Co-Supervisor : Assoc. Prof. Dr. Mine Özkâr

July 2010, 132 pages

Computational modeling of part relations of shapes is a challenging problem that has been addressed by many researchers since sixties. The most important source of the difficulty is the continuous nature of shapes, which makes the expression of shape very difficult in terms of discrete parts. When discrete parts are combined, they fuse and yield new parts, i.e. parts emerge. There is a number of methods that support emergent part detection. However all of these methods are based on strong assumptions in terms of what constitute a part. There is a need for a generic solution that treats a shape independently of any restriction resulting from analytical, geometrical, or logical abstractions. To this end, we have developed two novel strategies, which can be used both separately and jointly. Both strategies are relatable to the algebraic formalization of shape grammars (by Stiny). In the course of this thesis work, we have introduced a novel data structure called Over-Complete Graph to address the problem of part embedding in the existence of discrete registration marks; and we have developed a novel and robust method for the automatic selection of registration marks. Both developments are certainly useful for other visual problems. On the application side, we have tested our techniques on puzzling Seljuk patterns (from Kayseri) to demonstrate how the developed techniques give way to computational creativity.

Apart from the techniques we have developed, the most important contribution of our work is that shapes are treated as perceived wholes rather than composed, as compellingly demonstrated by Seljuk pattern experiments.

Keywords: part relations of shapes, emergent part detection, shape grammars, computation with shapes, shape algebras.

ÖZ

ŞEKİL GRAMERLERİ İÇİN PARÇA GÖMME

Yalım Keleş, Hacer
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Sibel Tarı
Ortak Tez Yöneticisi : Doç. Dr. Mine Özkâr

Temmuz 2010, 132 sayfa

Parça ilişkilerinin hesaplamalı modellenmesi, altmışlardan bu yana birçok araştırmacının ele aldığı zorlu bir problemdir. Zorluğun ana kaynağı şeklin devamlılık gösteren yapısıdır; bu yapı, şekli ayrık parçalar cinsinden ifade etmeyi zorlaştırır. Ayrık parçalar birleştirildiğinde kaynaşır ve yeni parçalar açığa çıkar, yani parçalar belirir. Beliren parçaları saptama desteği veren az sayıda metot vardır. Fakat bunların hepsi bir parçayı ne oluşturur konusunda güçlü varsayımlara dayanır. Bir şekle tüm analitiksel, geometrik veya mantıksal soyutlamalardan kaynaklanan kısıtlardan arınmış olarak yaklaşan, genel bir çözüme ihtiyaç vardır. Bu amaçla, hem ayrı olarak hem de bir arada kullanılabilecek iki yeni yaklaşım geliştirmiş durumdayız. Her iki yaklaşım da (Stiny'nin tanımladığı) şekil gramerlerindeki cebirsel biçimlendirmelerle ilişkilendirilebilir. Bu tez çalışması kapsamında, Aşırı-Tam Çizge adını verdiğimiz yeni bir veri yapısını ayrık kayıt işaretlerinin varlığında parça gömmeyi ele almak üzere tanımladık ve kayıt işaretlerinin otomatik seçimi için yeni ve güçlü bir yöntem geliştirdik. Her iki yöntem de diğer görsel problemler için kuşkusuz kullanışlıdır. Uygulama tarafında, tekniklerimizi (Kayseri'den) karışık Selçuklu desenleri ile test ettik ve geliştirdiğimiz tekniklerin hesaplamalı yaratıcılığa nasıl olanak sağlayacağını gösterdik.

Geliştirdiğimiz tekniklerin yanı sıra, bu çalışmamızın en önemli katkısı şekillere, zorlu Selçuklu motifleri deneyleri ile gösterildiği gibi, bileşimler olarak değil algılanan bütünler olarak muamele edilmesidir.

Anahtar Kelimeler: şekillerde parça ilişkisi, beliren parça tespiti, şekil gramerleri, şekil ile hesaplama, şekil cebirleri.

To my family

ACKNOWLEDGMENTS

It is a pleasure to thank to my supervisor, Prof. Dr. Sibel Tarı and my co-supervisor Assoc. Prof. Dr. Mine Özkâr, whose guidance, advice, criticism and encouragements from the beginning to the end enabled me to develop an understanding of the subject. This work would not have been possible without their support.

I would also like to thank to Prof. Dr. Ruşen Geçit and Assoc. Prof. Dr. Halit Oğuztüzün for their valuable comments and encouragements during the thesis monitoring meetings.

I would also like to express my gratitude to my husband, Fatih Keleş, for his endless support and love throughout this research.

Finally, I would like to thank to the Scientific and Technological Research Council of Turkey for the grants provided to this research through the project 108E015.

TABLE OF CONTENTS

ABSTRACT iv

ÖZ vi

ACKNOWLEDGMENTS ix

TABLE OF CONTENTS x

LIST OF FIGURES xiii

CHAPTERS

1 INTRODUCTION 1

  1.1 Background 5

    1.1.1 Embedding Relation 6

    1.1.2 Properties of U$_{ij}$ Algebra 6

    1.1.3 Shape Rule Application 7

    1.1.4 Part Relation 8

  1.2 Previous Works 10

  1.3 Motivation and Contributions 15

  1.4 Thesis Overview 17

2 EMBEDDING SHAPES WITHOUT PREDEFINED PARTS 19

  2.1 Shape Representation 20

    2.1.1 Mapping U$_{12}$ to V$_{02}$ 21

    2.1.2 Attributed Undirected Graph Representation 23

  2.2 Shape Graphs 25

    2.2.1 Equivalence Relations 25

    2.2.2 Part Relations 27

  2.3 Over-Complete Graph Representation 29

  2.4 Embedding Parts 33

2.5 The Two Step Part Embedding Algorithm 35
2.5.1 First Pass 35
2.5.2 Second Pass 41
2.6 The Reference Shape 42
2.7 Sample Implementation 45
2.8 Summary 50
3 WEIGHTED SHAPES FOR EMBEDDING PERCEIVED WHOLES 52
3.1 Mapping U_{12} to Weighted U_{22} 53
3.2 Embedding In Weighted U_{22} 55
3.3 The Genetic Algorithm Implementation using MATLAB Toolbox . . 56
3.4 Advantages of Using Weights In Embedding 59
3.5 No Presumptions About Shapes 61
3.6 Experiments And Discussion 68
3.6.1 Embedding for Visual Rule Based Construction 68
3.6.2 Embedding within Planes 72
3.6.3 Embedding Implicit Parts 80
3.7 Summary 84
4 CRITICAL POINT DETECTION IN DIGITAL LINE DRAWINGS 86
4.1 Introduction 86
4.2 The Edge Strength Function ([71]) 89
4.3 Detection of Critical Regions 90
4.3.1 Ellipse Shaped Analysis Windows 90
4.3.2 Reference Analysis Windows 92
4.3.3 Critical Regions 94
4.3.3.1 Region Analysis: First Phase 95
4.3.3.2 Region Analysis: Second Phase 98
4.3.4 Convergence Criteria 100
4.4 Critical Point Detection 101
4.5 Experiments and Discussions 101
4.6 Summary 112

5 CONCLUSION 113
    5.1 Future Works 115

REFERENCES 117

APPENDICES

A Sample Shape Embeddings 122
    A.1 Linear Sub-shape Embedding GUI Samples 122
    A.2 Generalized Sub-shape Embedding GUI Samples 126
B Implementation of the Edge Strength Function 130

VITA 130

LIST OF FIGURES

FIGURES

Figure 1.1 A sample visual rule [69]. 2

Figure 1.2 A sample shape ([64]). 2

Figure 1.3 (a) The embedded small square is shown with bold lines, (b) The rule in Figure 1.1 is applied. 3

Figure 1.4 (a) The embedded part is depicted with bold lines. (b) The resultant shape after the application of the same visual rule. 4

Figure 1.5 (a) The embedded part is depicted with bold lines. (b) The resultant shape after the application of the same visual rule. 4

Figure 1.6 Partial order in part relation is depicted for a shape in U_{02} [69]. 8

Figure 1.7 A sample U_{12} shape ([64]). 9

Figure 1.8 Some parts of the shape given in Figure 1.7 ([64]). 9

Figure 1.9 A sample shape in U_{22} ([64]). 9

Figure 1.10 Some parts of the shape given in Figure 1.9 ([64]). 10

Figure 2.1 A sample line shape. 20

Figure 2.2 Topologically critical points are labeled. Apart from the two intersection points $\{p_3$ and $p_7\}$, all the remaining points are the boundary points of the maximal lines. 21

Figure 2.3 (a)Shape in $U_{12}.V_{02}$ with labels for line segments and points shown for illustrative purpose, (b)The corresponding shape in V_{02} obtained with a particular mapping. 23

Figure 2.4 Sample node-to-node connection angles between the nodes (n_i,n_j) and (n_i,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): $(\alpha_2 - \alpha_1)$. It is computed via node-to-node connection angles when necessary. 24

Figure 2.5 Sample shape for a generalized case. Linear or not, segments of U_{12} elements, have a boundary elements set in V_{02}. 25

Figure 2.6 Three different instances that are shape equivalent under Euclidean transformations. 26

Figure 2.7 Sample shape (a) to be detected in shape (b) 27

Figure 2.8 Two subgraphs for the shape graph of the Figure 2.7 28

Figure 2.9 A subgraph of the graph of the shape in Figure 2.7-(b) 28

Figure 2.10 The implicit connection between the nodes n_i and n_j are made explicit in the over-complete representation of this shape. 30

Figure 2.11 (a) Sample shape with labelled fragments. (b) Corresponding V_{02} shape . . 31

Figure 2.12 (a) Concatanation of f_1 and f_2 forms a perceptual whole. (b) Node relations through $f_1 f_2$ segment is reflected to V_{02} shape. 31

Figure 2.13 (a) Concatanation of f_4 and f_3 forms a perceptual whole. (b) Node relations through $f_4 f_3$ segment is reflected to V_{02} shape. 32

Figure 2.14 (a) Simple shape graph of the shape in Figure 2.11-(a). (b) Over-complete graph of the same shape 32

Figure 2.15 (a) Simple graph of the shape in Figure 2.5. (b) Over-complete graph of the same shape. 33

Figure 2.16 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. 34

Figure 2.17 Four exemplary cases to discuss the part embedding algorithm 35

Figure 2.18 Example rule for *Case 1* is displayed for four sample mappings $(1-4)$. The candidate mappings are displayed in the first row. A 3-node source subgraph is selected from the source graph and shown on the left 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 indices. 37

Figure 2.19 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*. In the figure above, the source graph nodes ${n_1}^S, {r_1}^S, {r_2}^S$ are labeled as n_{s1}, r_{s1}, r_{s2} respectively. 39

Figure 2.20 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 the source shape is a part of the target shape, while (3) and (4) are rejected mappings in the second pass of the algorithm. In the figure above, the source graph nodes ${n_1}^S, {r_1}^S, {r_2}^S$ are labeled as n_{S1}, r_{S1}, r_{S2} respectively. 40

Figure 2.21 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, which is displayed with light-weighted lines in the rule definition. Since all the nodes in a reference shape graph is of type NF, 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. 41

Figure 2.22 Application of the same rule in Figure 2.21 with a different reference shape. 44

Figure 2.23 Reference shapes can be defined in any desired way. Some sample reference shapes for the same rule in Figure 2.21 are displayed with bolder lines. . . . 44

Figure 2.24 Sample Canonical representations for the curved line segments $s2$, $s4$ and $s6$ of the shape in Figure 2.5 and the straight line segment $l4$ in Figure 2.3. All canonical representations of straight line segments are similar. The only difference between two such representations may be in the density of the plotted points. . . 47

Figure 2.25 Screenshot of a sample embedding within a shape with curved components. 48

Figure 2.26 Screenshot of a sample embedding within a linear shape. 49

Figure 3.1 (a) A U_{12} shape, (b) Corresponding weighted U_{22} shape. 54

Figure 3.2 Part Embedding GUI 57

Figure 3.3 GA Profiler GUI 58

Figure 3.4 The options provided with the GA Profiler GUI for the cross-over function in addition to the Heuristic method are Two-Point, Scattered, Intermediate and Single-Point cross-over functions. 59

Figure 3.5 Part alignments of 10 successive iterations of the GA without using weights. Except for the iterations (5) and (7), the alignments are far from expected results. In these experiments the following GA parameters are used: population size = 150, number of generations = 70, scale range = 0.9-1.4, transformation range = [-70,70], rotation range = [0,359], cross-over fraction= 0.6, elite count = 2. 60

Figure 3.6 Part alignments of 10 successive iterations of the algorithm with weighted images. Except from experiment (5), all the alignments are very close to the one of the expected solutions. In these experiments the following parameters are used in GA: population size = 150, number of generations = 70, scale range = 0.9-1.4, transformation range = [-70,70], rotation range = [0,359], cross-over fraction= 0.6, elite count = 2. 61

Figure 3.7 For a shape which is composed of a curve with a highly free-form nature and a straight line, the proposed algorithm makes successful embeddings regardless of the types of its basic elements (i.e. line or curve). 62

Figure 3.8 The sample shapes are composed of straight lines and curves connected end to end. The estimated four transformation parameters, one for orientation, one scale, two for transformations (in x and y coordinates) with the GA for (a) is (330, 1.25, -1.7, 34.2); for (b) is (300, 0.98, 13.4, 11.7) and for (c) is (329,0.99,-2.16,19.4). 63

Figure 3.9 All different part embeddings that are obtained with the proposed algorithm is depicted for two sample curved part shapes. 64

Figure 3.10 All different part embeddings that are obtained with our method is depicted for four sample part shapes with straight lines. 65

Figure 3.11 Sample embeddings of a set of different parts within the same initial shape. 67

Figure 3.12 Embedding a point using a reference shape 69

Figure 3.13 Two sample embeddings. 70

Figure 3.14 The visual rule which when applied combines two points with a line. . . . 70

Figure 3.15 (a)The shape obtained after a number of rule applications to the shape in Figure 3.1, (b) Corresponding weighted U_{22} shape. 71

Figure 3.16 Sample part embeddings. 72

Figure 3.17 Embedding a shape that is composed of parts of triangles into the line crowded Seljuk pattern. 73

Figure 3.18 A sample part embedding where the initial shape is composed of planar elements in addition to lines. 73

Figure 3.19 Sample part embeddings. 74

Figure 3.20 A Seljuk pattern from Huand Hatun Tomb (1237,1238), Kayseri, Turkey (drawn by Assoc. Prof. Dr. Mine Özkâr). 75

Figure 3.21 The Seljuk pattern in Figure 3.20 is redrawn with thick lines which are generated considering the weaving of the stripes. 76

Figure 3.22 (top) Embedding results of 100 distinct GA runs are aggregated, where the searched part is a one pixel-wide quadrilateral shape, (bottom) Embedding results of 30 distinct GA runs are aggregated, where the searched part is a six pixel-wide quadrilateral shape. The initial shape in both experiments is the planar representation of the Seljuk pattern depicted in Figure 3.21. 77

Figure 3.23 Each figure is formed by aggregating the embeddings of multiple runs. The number that is shown below each sub-figure stands for the index of the number of aggregated embeddings. 78

Figure 3.24 Aggregate embeddings of 100 runs for embedding a rhombus shape within the initial Seljuk pattern. 79

Figure 3.25 Aggregate embeddings of 20 runs for embedding a pentagram shape within the initial Seljuk pattern. 80

Figure 3.26 Sample implicit part embeddings. 81

Figure 3.27 Sample part embeddings. 82

Figure 3.28 Sample star-shape embeddings within the tile. The obtained embedding result on the right-most window of the GUI is the result of 25 different GA iterations. GA parameters: number of population=100, scale range=[0.95, 1.1], position range = [-200,+200], orientation range=[0,359], number of generations 30.. 83

Figure 3.29 (left) Embedding a thin ten-legged star shape of one pixel-width; (right) Embedding a thick ten-legged star shape of six pixel-width. The initial shape is a part of the resultant shape obtained after the experiment shown in Figure 3.28. . . 84

Figure 4.1 Selected set of shapes. 89

Figure 4.2 Corresponding edge strength functions of the shapes in Figure 4.1. 90

Figure 4.3 Sample ellipse window alignments 91

Figure 4.4 Gradient direction (d_g) and the ellipse alignment direction (d_a) for a sample ellipse window. 91

Figure 4.5 The reference shape and ellipse windows in different dimensions (KMV: window mean value): (a) ellipse minor ax. dim.: 15 px; major ax. dim: 30 px; KMV: 0.5087, (b) minor a.d.: 20 px; major a.d.: 40 px; KMV: 0.4239, (c) minor a.d.: 30 px; major a.d.: 60 px; KMV: 0.3121, (d) minor a.d.: 40 px; major a.d.: 80 px, KMV: 0.2433. 92

Figure 4.6 Distribution of the Reference Window Mean Values. 93

Figure 4.7 Sample critical regions with different intrinsic properties. 94

Figure 4.8 Effective range of ellipse axis dimensions and the reference window mean distribution, which is computed with *rho*=8. 95

Figure 4.9 (a) Critical regions, (b) Merged critical regions. 98

Figure 4.10 Distribution of the field mean values over varying disc dimensions. 99

Figure 4.11 Sample shapes for the experiments. 102

Figure 4.12 The candidate critical regions, shown in bold lines, are determined after the first phase of the analysis. The number below each shape stands for the iteration index. 102

Figure 4.13 The merged critical regions obtained after the second phase of the analysis. The numbers stand for the iteration indices. The result depicted as the 13'th iteration is the converged result. 103

Figure 4.14 The candidate critical regions determined after the first phase of the analysis. This sample is converged after 4 iterations. 104

Figure 4.15 The merged critical regions obtained after the second phase of the analysis. 104

Figure 4.16 The candidate critical regions determined after the first phase of the analysis. This sample is converged after 7 iterations. 105

Figure 4.17 The merged critical regions obtained in the second phase of the analysis. . 105

Figure 4.18 A different reference shape. 106

Figure 4.19 The shape in Figure 4.1-(a) is redrawn with the same brush that is used in constructing the reference shape in Figure 4.18. 106

Figure 4.20 The first 4 iterations of the analysis for the shape in Figure 4.19. 107

Figure 4.21 Critical regions determined for the cluttered image shown in Figure 4.20. . 108

Figure 4.22 Critical regions depicted with circles. The field is computed with (ρ=8). . . 108

Figure 4.23 Critical regions depicted with circles. The field is computed with (ρ=8). . . 109

Figure 4.24 Data set created similar to the ones in ([73]). (a) Semicircle-shaped curve, (b) Chromosome-shaped curve, (c) Infinity-shaped curve. 109

Figure 4.25 Approximation of the chromosome data with the proposed method. 110

Figure 4.26 (a) The proposed method, (b) Rosenfeld-Johnston algorithm, (c) Rosenfeld-Weszka algorithm, (d) Freeman-Davis algorithm, (e) Sankar-Sharma algorithm, (f) Anderson-Bezdek algorithm, (g) Teh-Chin algorithm (k-cosine), (h) Teh-Chin algorithm (k-curvature),(i) Tel-chin algorithm (1-curvature). All the sample results from (b) to (i) are copied directly from [73]. 111

Figure A.1 One of the four embedding results for the examplary case in Figure 2.17-(a). 122

Figure A.2 One of the two embedding results for the examplary case in Figure 2.17-(b). 123

Figure A.3 One of the two embedding results for the examplary case in Figure 2.17-(b). 123

Figure A.4 One of the four embedding results for the examplary case in Figure 2.17-(c). 124

Figure A.5 One of the four embedding results for the examplary case in Figure 2.17-(c). 125

Figure A.6 One of the eight embedding results for the examplary case in Figure 2.17-(d), with the user defined constraint. 125

Figure A.7 One of the eight embedding results for the examplary case in Figure 2.17-(d), with a different user defined constraint. 126

Figure A.8 One of the eight embedding results for a sample shape. 127

Figure A.9 One of the eight embedding results for a sample shape. 127

Figure A.10 One embedding results for a sample shape. 128

Figure A.11 One embedding results for a sample shape. 129

CHAPTER 1

INTRODUCTION

In conventional shape recognition approaches, for representation purposes, shapes are decomposed into parts according to some rules. In their seminal paper, Hoffman and Richards argue that in the visual system a shape is decomposed into a hierarchy of parts and this decomposition is based on the regularity of nature ([24]). For this purpose, the regularity is defined with the well known *minima rule* on the boundaries of parts. Accordingly, a plane curve is decomposed at points of the negative curvature minima. When the problem is the recognition or the matching of shapes, standardization of the representations of shapes and hence fixing the definitions of part boundaries is necessary and beneficial. However, in design activity such restrictions in the representation of shapes - sticking to predefined decomposition of shapes - may inhibit creativity. In this domain, instead of eliminating ambiguities within shapes via unique interpretations of shapes, exploiting them to their limit is considered as real sources of creation. As a result, different computational approaches are necessary for representing shapes to support creative design activities.

In creative thinking, regularities within forms are not usually the preferred sources of creation. Most of the creative works/arts are being generated by destroying the rules/biases imposed on the relevant subject. The shape grammar formalism which was first introduced by Stiny and Gips in 70's ([59]), provide the necessary formal devices for this purpose when the relevant subject is a shape. The formal way of pictorially modifying shapes as a method of exploiting the ambiguity is established via well defined algebras. The formalism has been developed significantly by George Stiny since its introduction. A recently published book of Stiny ([69]) presents the shape grammar formalism both with respect to its philosophical and formal aspects. The work inspired us while pursuing the computational problem we argue in this thesis.

In the shape grammar formalism, shapes are operated via visual rules. The philosophy behind the rule application is modifying a given shape pictorially. The grammars in this context are not symbolic as opposed to the other grammar systems, like grammars defined in Chomsky hierarchy in formal language theory. A sample shape rule is specified pictorially in the form of $A \rightarrow B$ as shown in Figure 1.1.

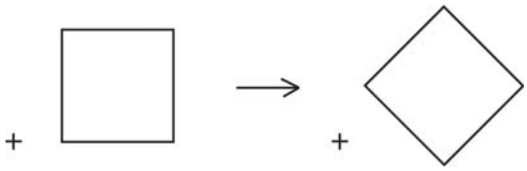


Figure 1.1: A sample visual rule [69].

In the rule $A \rightarrow B$, A stands for the left hand side shape of the rule and B represents the right hand side of the rule. Any given left hand side shape of a rule can act either as a non-terminal or a terminal symbol. The shape acts as a *non-terminal* when an occurrence of it in the initial shape is replaced with the right hand side shape, on the other hand the same shape acts as a *terminal* when it is left unchanged in the resultant/terminated shape. Although a shape can be interpreted either as a terminal or a non-terminal, shapes are not symbols which stand for some entities. Instead they are composed of geometric entities like points, lines, planes, and so on.

Assume that an initial shape is created with two overlapping squares as shown in Figure 1.2. In most computer aided design tools or vector supported drawing tools, when a shape is created with such (predefined) primitives, a commitment has already been made on the parts of the shape. Such a commitment restricts the kind of modifications applicable to the shape. For the given initial shape only the two squares are accessible using these tools although there are many other parts that can be perceived within the shape.

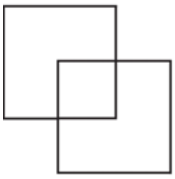


Figure 1.2: A sample shape ([64]).

In the shape grammar formalism, parts of shapes are not predefined. What one can see in a shape is accessible through visual rules, independent of how the shape is created/defined. The decomposition of a shape is not fixed in advance which would otherwise restrict the selectable parts within the shape. For the initial shape in Figure 1.2, the application of the sample rule in Figure 1.1 may result in a shape as shown in Figure 1.3-(b). This new shape is obtained by embedding the left hand side of the rule to the small square that emerges as a result of the geometric arrangement of the two big squares (Figure 1.3-(a)). Although this small square is not made on purpose, the shape grammar formalism allows us to access such emergent parts.

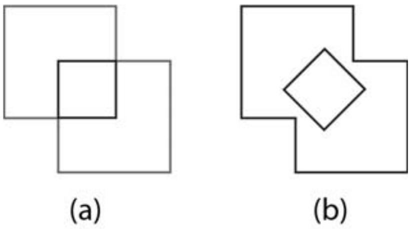


Figure 1.3: (a) The embedded small square is shown with bold lines, (b) The rule in Figure 1.1 is applied.

Rule application process in shape grammars is recursive. Each rule application results in a new shape to work with. The dynamics of the selection of a part (e.g. the square drawn bolder in (a) above) to apply a rule is another research topic that is left beyond the scope of this thesis. In this work, it is assumed that an external subject decides about the specific part to apply a rule to. Any such decision creates a new way while working with shapes. For example, application of the same rule to the same initial shape, while embedding a different part this time, (which is shown with bold lines in Figure 1.4-(a)) results in a totally different shape (Figure 1.4-(b)).

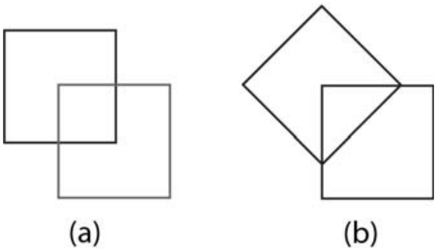


Figure 1.4: (a) The embedded part is depicted with bold lines. (b) The resultant shape after the application of the same visual rule.

If we continue to apply the same rule to the resultant shape, while embedding the square shown in bold lines in Figure 1.5-(a), we would obtain the shape in Figure 1.5-(b).

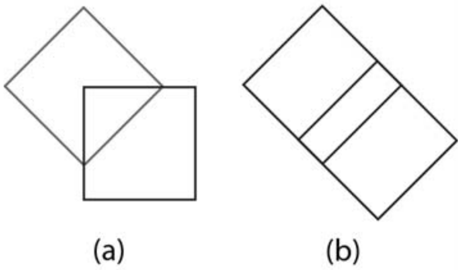


Figure 1.5: (a) The embedded part is depicted with bold lines. (b) The resultant shape after the application of the same visual rule.

In summary, visual rules are utilized for acting on shapes in the shape grammar formalism. A shape is not composed of predefined units but is temporarily decomposed into some parts according to the desired action via visual rules. Since rules are visually defined, any perceived part of a shape is accessible. The recognition of any part within a given shape without a predefined decomposition is a fundamental problem to be resolved for the implementation of the visual computing.

Most of the existing computer implementations of shape grammars focus on the rule-based recursion and overlook the notion of part relations and embedding problem. The ones which deal with the embedding problem provide partial solutions by making predefined geometrical abstractions related with the primitives within shapes in advance. Based on these observations the fundamental problem of handling part relations of shapes coherent with the shape grammar formalism initiated the basis of the research provided in this thesis.

Before going into the detailed discussion about the proposed solutions for this problem, let us give the outline for the rest of this chapter. The introductory material that are necessary to follow the work defined in the context of this research is provided in the upcoming section. Following that, the summary of the related literature is provided. Next, the motivation and contributions of this research is discussed. Finally, an overview of this document is provided in the last section.

1.1 Background

The shape grammar formalism provides all the necessary tools for any kind of modifications like decomposition of shapes or generation of new shapes. The formal devices for any kind of modifications through visual rules are characterized by George Stiny with shape algebras. In this section, a very brief introductory material is provided about shape algebras with embedding and part relations of shapes.

Algebras of shapes[1] have three main constituents: shapes (the basic elements of shapes and the space they are represented in), part relations for shapes that include Boolean operations and Euclidean transformations.

Shapes are composed of basic elements of different dimensions, i.e. points, lines, planes and solids, or any combination of these. They are represented in Euclidean space. Each U_{ij} algebra is defined according to the dimension of the basic elements in a shape and the dimension of the Euclidean space they are represented in. In this representation, i stands for the dimension of the basic elements of shapes and j is for the dimension of the space in which basic elements are combined. For example, U_{12} contains shapes of lines (i=1) in two dimensional Euclidean Space (j=2).

In order to understand the properties of each algebra, embedding properties of the basic elements should be understood.

---

[1] Reader may refer to [69] to deeply elaborate the subjects summarized in this section.

1.1.1 Embedding Relation

Embedding relation is defined in shape grammar formalism as a partial order which has the following properties:

1. *reflexive*: every basic element is embedded within itself,
2. *antisymmetric*: two basic elements are identical whenever first is embedded in the second and the second is also embedded in the first,
3. *transitive*: for three basic elements a,b and c, if a is embedded in b and b is embedded in c, then a is embedded in c.

Embedding relation is identity for the basic elements of type points. In other words, a point is embedded within another point as an atomic unit in a unique way. On the other hand, only the same kind of basic elements can be embedded within another basic element that has dimension greater than zero. For example, only lines can be embedded within lines or only planes can be embedded within planes, and so on. In contrast to points, embedding is not identity for these basic elements; there are infinitely many possible embedding options. This is the result of avoiding predefinitions for shapes. In other words, shapes with these basic elements are not defined with fixed units which otherwise would limit the number of parts that can be embedded in visual computations. In all geometric modeling tools such as computer-aided design (CAD) tools or vectorial drawing tools, shapes/forms are defined using some set of primitives. For example, when a rectangle shape is drawn with a rectangle primitive provided in the tool set, accessing the individual lines of the rectangle is not possible. Embedding relation defined in the shape grammar formalism is free from such restrictions. Any part of a given shape is accessible regardless of the shape's creation history.

1.1.2 Properties of U_{ij} Algebra

U_{ij} algebras carry most of the properties of a Boolean algebra, but except from U_{00}, all fail to be one due to the nonexistence of a unit and complement. These algebras carry the properties of a Boolean ring with an empty shape and no unit. The only Boolean algebra of shapes is U_{00}. In U_{00} there are exactly two shapes, namely the empty shape and the shape with just one point, corresponding to *false* and *true* in the Boolean algebra respectively.

In the shape grammar formalism, in addition to the simple shape algebras, composite algebras can be defined as the Cartesian products of different U_{ij} algebras. In composite shape algebras, basic elements of different types are arranged in multiple spatial dimensions. In that case, each type of basic element is treated as if it exists in a different operational layer. For example $\mathrm{U}_{02}.\mathrm{U}_{12}$ algebra is a composite algebra which contains shapes of points and lines together in the two dimensional Euclidean space. All the Boolean operations (sum, difference and products) between two shapes - both belonging to the same composite algebra - are performed among the same type of shape elements of the two shapes.

The U_{ij} algebra is extended with the label and weight algebras, V_{ij} and W_{ij} respectively [64]. Basic elements in U_{ij} algebra are associated with labels selected from a vocabulary and form the corresponding V_{ij} algebra. Labeling shape elements may serve to different purposes. For example, one such usage could assign certain semantic meanings to the associated shape elements. Similarly, basic elements of a shape in U_{ij} algebra can be assigned some weights and corresponding shape is then operated in the weight algebra (W_{ij}). For example, in W_{12}, lines may have thickness or in W_{02}, points may have areas as weights. Both label and weight algebras keep the Boolean properties of the corresponding U_{ij} algebra. Composite algebras can also be formed with the Cartesian products of different U_{ij}, V_{ij} or W_{ij} algebras.

1.1.3 Shape Rule Application

In the shape grammar formalism, any action on shapes is defined by means of shape rules. Shape rules provide a general mechanism to operate with shapes independent of the algebra they belong to. The rules are applied recursively regarding the part relation, the Boolean operations and Euclidean transformations. Given an initial shape C and a set of shape rules in the $A \rightarrow B$ form (Figure 1.1), the application of a rule is done by these three steps:

1. Find a Euclidean transformation Γ so that $\Gamma(A)$ is equivalent to a part of C
2. Define a new shape C', by $C' = C - \Gamma(A)$,
3. Obtain the final shape C'', by $C'' = C' + \Gamma(B)$

As a result, the final shape after applying the rule $A \rightarrow B$ to the shape C is $C'' = C - \Gamma(A) + \Gamma(B)$.

1.1.4 Part Relation

Before defining the part relation, let us mention the maximal representations of shapes in the shape grammar formalism. As it is stated before, shapes with basic elements of dimension greater than zero can be represented in infinitely many different ways each corresponding to a different decomposition. Many different representations of such shapes may belong to the same shape. In order to identify distinct shapes and explain the spatial properties in an unambiguous fashion, the maximal representation provides a unique canonical way.

A maximal element in a shape is the basic element in the finite set of basic elements of the shape, that can only be embedded within itself. Accordingly, a maximal representation of a shape is the set of smallest number of basic elements that combine to form the shape.

A shape *A* is *part of* another shape, let us call it *B*, if every maximal element of the shape *A* is embedded in a maximal element of *B*. Part relation is also a partial order on shapes. It can be used to form lattices for shapes. A sample distributive lattice for a shape composed of three points shown in Figure 1.6 depicts the partial order of part relations.

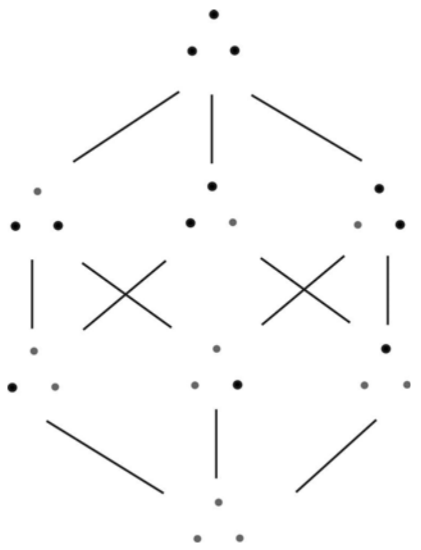


Figure 1.6: Partial order in part relation is depicted for a shape in U_{02} [69].

The lattice in Figure 1.6 shows the eight possible parts of a shape with three isolated points together with how the parts are related. This diagram also displays that a shape and its parts in U_{02} form a complete Boolean algebra of a finite set and its subsets.

A shape in U_{0j} algebra contains discrete elements, hence its possible decompositions are finite and countable. However, for shapes in the U_{ij} algebras, where *i* is different from zero, possible decompositions are infinitely many. For example, a line in finite length can be decomposed into a finite number of line segments; each can also be decomposed into a finite number of sub-segments within itself. In this way, a line can be decomposed into lines in infinitely many different ways. For the U_{12} shape given in Figure 1.7 which is composed of eight maximal lines as its basic elements, a subset of its parts is shown in Figure 1.8. This set, however, does not exhaust all parts of this shape. There are infinitely many of them.

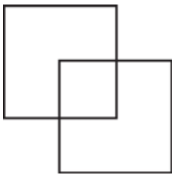


Figure 1.7: A sample U_{12} shape ([64]).

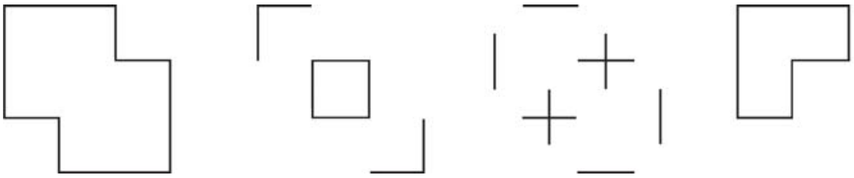


Figure 1.8: Some parts of the shape given in Figure 1.7 ([64]).

Similarly, for a shape in U_{22} shown in Figure 1.9, a subset of its parts are depicted in Figure 1.10.

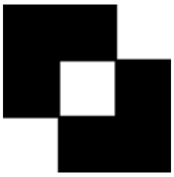


Figure 1.9: A sample shape in U_{22} ([64]).

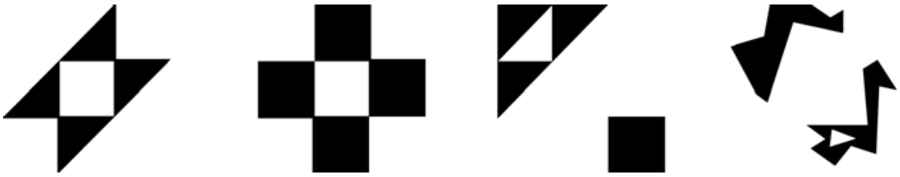


Figure 1.10: Some parts of the shape given in Figure 1.9 ([64]).

Any particular decomposition for such shapes can also be represented with the distributive lattice to resolve the parts and the relations of parts by Boolean sums and products.

For shapes which belong to U_{ij} algebra ($i, j \neq 0$), possible decompositions are not predefined. In the shape grammar formalism, each decomposition reveals different parts of shapes that would be useful for visual computations via shape rules. Independent from distinct creation histories, a shape can be decomposed into parts in many different ways according to what one wants to see within a shape. By this way, different reinterpretations of shapes become possible which is the power behind the shape grammar formalism.

1.2 Previous Works

The pioneering work of George Stiny and James Gips [59] initiated a new research field referred to as shape grammars. The shape grammar formalism is then extended substantially by Stiny [62], [63], [64], [65], [66], [68]. Numerous other shape grammars and their implementations also made important contributions to the field that is of significant interest to the computational design community. Stiny recently reinstated the philosophical and technical characteristics of the theory of shape grammars providing a renewed reference point for this field [69].

As it is emphasized in the previous section, the power of shape grammar formalism comes from its ability to provide reinterpretation of shapes. It is possible only if shapes are represented in such a way that any decomposition is made possible. Although it is straightforward with pencil and paper, computationally this is a challenging problem. It is due to the fact that computational platforms work with symbol manipulation. Yet, a good shape grammar interpreter should be deprived of symbols or at least, it must behave like that.

Early shape grammars were defined for analyzing arts on paper [59],[67],[21]. Following these initial works, shape grammars are widely used in architectural plans and some historical structures. Contributing largely to the design research, many of these works in the literature used shape grammars to identify and formalize the design styles of the previously generated designs and generate new designs with the same style. One of the first examples of shape rules for generating designs of a given style is Stiny's Chinese ice-ray lattices [60]. Following that, in architecture Palladian style villas [61], Frank Lloyd Wright's Prairie Houses [33], Queen Anne Houses [19]; in furniture design Hepple-white style chair backs [29]; in art work on deStijl trend [30] may be given as examples to the use of shape grammars in this respect.

Many of these shape grammars are not computationally implemented while the existing implementations are hard-coded set grammars. In set grammars, discrete symbols, like labels, are used to locate parts in an initial shape with some combinatorial methods, in which case the advantage of emergent part detection is lost.

Later, the potential of shape grammars in the field of engineering design is realized by many researchers and many different shape grammars are developed in this field even in some commercial product designs. Although a subset of these works is implemented, the existing implementations are mostly application specific set grammars and provide partial solutions to the more general problem of computational part embedding.

Many of the engineering shape grammars are equipped with a set of parametric attributed labeled shapes to grasp the interaction between function and form (i.e. Lathe Grammar [10] and Robot Arm Grammar [76]). The Coffeemaker grammar [1] is an example to consumer product design, which is implemented as a web-based Java applet, is hard-coded according to the specified rules. It does not operate considering the geometry for embedding but depends on extensive usage of labels. Other similarly hard-coded grammars are MEMS grammar [2], which is based on labels, and space truss grammar [58] in which only triangles are used. Neither of these grammars takes advantage of emergent parts. Similarly, Pugilese and Cagan employed symbols in matching parts in their implementation of the motorcycle grammar [53]. All these works are implemented as set grammars in which part matching is performed by comparing the labels common in both shapes rather than considering the geometric properties of shapes. A broad summary of the shape grammars in engineering design can be found in [11].

There are some implementations for a class of shape grammars referred to as basic shape grammars ([32]). These types of grammars focus on addition rules, and their sequences, but not the technicalities of emergent shapes ([46], [75]). The purpose of developing Shaper2D [46] is to facilitate shape grammar learning. This tool is not designed for professional design generation; it provides an alternative method to encourage students to learn basic concepts of the shape grammar theory. The set of shapes is limited to a palette of pre-programmed shapes: squares, rectangles and triangles. Similarly, 3DShaper [75] is restricted to shapes of pillars, oblongs or cubes.

Heisserman developed a solid grammar interpreter, named Genesis, which represents solid shapes as graphs and used it in a generative grammar for Queen Anne houses [23]. In this interpreter a solid rule is specified with a set of match conditions and a sequence of operations. A logical reasoning mechanism is exploited to express complex geometric conditions and operations. In a rule application, a graph is modified only if it satisfies the conditions defined in the rule. In other words, part matching depends on the predefined constraints within the rules such as labels and some topological parameters rather than the geometry itself. The emergent part detection problem is not the issue of this work.

On the other hand, there are a few works that directly address the computational aspects of the part relations of shapes. These approaches define algorithms and data structures suitable for part embedding problem that could be used in the development of shape grammar interpreters.

McCormack and Cagan [44], [45], propose a parametric part detection method pointing out that the existing implementations for rule application are limited to certain rule sets in given engineering or design problems. Moreover, they also point out that the existing solutions to the part embedding problem transform the entire shape rather than its parts while embedding. They decompose shapes into parts which are hierarchically ordered based on their constraining relations with other parts [44]. Recognition is then performed by matching the shapes within the same subset and their combinations. Parametric shape recognition is especially useful for engineering shape grammars with well-defined problems. If the default hierarchy is not desired, the hierarchic grouping of parts can be constructed with the designer's intention. However, for each different hierarchy, the set of embedded parts would change; in which case the designer should be aware of which parts are missed and take the responsibility. Moreover, for the proposed solution for curved shapes [45], since the matching of the curves depends on the computed transformations in the first pass via straight line representation of the shape, this may result in losing some emergent parts even with the default hierarchy.

Ramesh Krishnamurti focused on the development of suitable data structures and algorithms to solve the computational part embedding problem for line shapes [35] which is the most prominent approach that is proposed for line shapes. It is based on the maximal element representation. In this representation, a shape is defined by a set of maximal lines each of which is described by its carrier and the boundary points [34]. Carrier of a line is the infinite line that the line is embedded in. In the provided solution, shapes are represented by ordered lists such that the set of maximal elements with the same carriers are put in the same list. As a result, sets of co-linear maximal lines are constructed and the lines in each set are ordered according to their boundaries.

Utilizing this maximal representation, all possible parts of an initial shape that is similar to a left hand side shape of a visual rule is computed via the *distinguishable points* together with the given *labeled points*. These points serve as reference points in matching and necessary, because there are infinitely many different transformations that embed a line segment into another under Euclidean transformations. Distinguishable points in a shape are defined as the intersection points of the maximal lines and their extensions. For each possible mapping of the distinguished points of the two shapes, a transformation is computed. Following that the descriptors and the boundaries of the two shapes are compared to validate the embedding relation between the two shapes. This representation is general enough to represent shapes which allow detection of parts within a shape, even the emergent parts.

There are three well known shape grammar implementations based on the maximal representation proposed by Krishnamurti that support emergent part detection [36], [12], [70]. Although the solution provided by Krishnamurti for part detection is a generic solution for shapes with straight lines, all these implementations work with rectilinear shapes. Tapia's implementation, GEdit, is the most advanced one which provides a graphical user interface for defining and applying visual rules. However, this application is also restricted to work only with the orthogonal shapes.

Apart from the solutions for line shapes, there are a few works that focus on defining part relations of shapes with curved elements [45], [14], [26], [27]. It is widely accepted that computationally defining the part relations of curved shapes are more challenging than that of line shapes. Some of the previous approaches explored the part detection problem considering some constraints related with the form of the shapes. Hence, with these constrained forms, they are far from providing a general solution for curved shapes.

Chau et al.'s shape grammar implementation is based on maximal representation which considers maximal circular arcs in planes (2D) and space (3D) under Euclidean transformation [14]. However, apart from the circular arcs, their solution does not include the shapes which contain more general free-form elements.

McCormack and Cagan's implementation supports free-form curves by converting freeform curves to the shapes composed of straight line segments [45]. As we mentioned in engineering shape grammars above, some of the embedding properties of the free-form curves are not considered in this work. Hence, this may result in losing some emergent parts.

In Jower's thesis, the shape grammar formalism is extended with new algebras that contain curves with more free-form nature [26]. A theoretical framework is constructed coherent with the George Stiny's shape algebra definitions and conventions. However, a general mechanism to define shape grammars for all types of free-form curves is not given. This is due to the reason that the properties of different types of curves is very different to generalize. Two example types of curves are selected to validate the theoretical construction proposed in his thesis; one for quadratic curves and the other is for cubic Bezier curves in the planar domain.

Recently, Jowers and Earl [27] have proposed an intrinsic matching algorithm for developing a shape grammar interpreter for shapes composed of parametric curve segments. Their method is based on the comparison of the intrinsic properties of two space curves which serve as descriptors of curve segments. This is analogous to the infinite lines that are used as descriptors of maximal line segments proposed by Krishnamurti. They pointed out however that intrinsic matching is not unique for shapes composed of geometric elements with constant intrinsic properties, such as circular arcs. For such shapes, it is necessary to find alternative methods for determining embedding properties, such as using Krishnamurti's *distinct points*. Again in this work, they have pointed out that providing a general method to be used for any types of curves is not easy. Actually, the parameters, λ and u, which are used for determining embedding relations of curve segments are dependent on the particular parametric curves used to compose a shape (i.e. Bezier curves).

Computational modeling of the part embedding problem is very challenging especially when (ambiguity rich) shapes are considered. As it is summarized above, there are only a few approaches that focus on solving this problem.

The solutions provided so far either assumed that (1) shapes are composed of straight line segments, so that the only shapes that are allowed in a shape grammar would be rectilinear shapes or (2) shapes are composed of curved elements which can be defined by fitting a higher order function. Each of these assumptions is restrictive in the sense that it forces a user/designer to work with pre-specified type of shapes.

Shapes are continuous entities and a generic solution to the part detection problem should be able to work with any shape without the need to define the shape with certain analytic abstractions. Fitting a linear form to a line segment and assuming that the shape is composed of linear shape elements everywhere is such an analytic abstraction. Any solution coherent with the shape grammar formalism should be capable of finding any occurrence of a given shape within another regardless of the types of basic elements it possesses. What we believe the most fundamental element of the shape grammar formalism is respecting the continuous nature of shapes by treating a shape as it is. Hence, a general solution should be free from such presumptions about shapes. In view of these, the works proposed in the context of this thesis provide two novel approaches to define generic solutions to the problem of part embedding.

1.3 Motivation and Contributions

The research on the problem of part embedding up until now has been directed with assigning some analytic abstractions to the basic elements of shapes. We think this is problematic since making presumptions related with the basic element types of shapes directly limits the allowed set of shapes that can be utilized in part embeddings. That is any shape that has a part violating the assumed form can not be used in part searches. We believe that, such restrictions contradict with the philosophy behind the shape grammar formalism.

The motivation of this work is providing a generic solution for the part embedding problem for line shapes of any kind in two dimensional Euclidean space. In this principle, two different approaches are proposed in the context of this study. Both approaches provide a pictorial solution to the problem considering the perceptual wholes instead of making analytical abstractions for shapes. This is consistent with the nature of shape modifications in the shape grammar formalism. In both solutions a set of constraints which are inherent in the given shapes are utilized via the well defined mappings to different algebraic levels.

In the first approach, a shape is mapped to a lower algebraic level so that a set of discrete points serves as registration marks and is utilized in the identification of possible embeddings. These points are the external constraints that are defined temporarily and imposed on a shape in order to utilize in part matching. As a generic case these constraints can be defined automatically by considering the intrinsic properties of shapes together with the shape's topological properties. In addition to this default set, additional external constraints can always be defined by a user/designer. The major contributions of this method can be listed as follows:

- A generic solution to emergent part detection problem is developed which works for any type of line shapes.
- Instead of an assumed geometric notion or symbolic representation, the proposed solution works with perceptual wholes.
- A new graph data structure, which we call over-complete graph, is proposed as a means for temporarily representing shapes. This data structure is equipped with the useful tools and algorithms to effectively traverse a shape during part searches and very suitable to represent shapes as a composition of perceived wholes.
- The notion of the reference shape is introduced to the literature to solve the nondeterministic type searches.
- In addition to the constraints that are inherent within the shapes, the user/designer is given the opportunity to constrain the part searches.

In the second approach, the set of constraints which determine the embedding properties of a shape is obtained from modeling the interactions inherent in the shape in a higher level algebra via weights. The contributions of this approach can be summarizes as:

- A generic solution to emergent part detection problem is developed which is based on an evolutionary approach.

- Weighted shapes in a higher level algebra is utilized to enable them as registration marks in part embedding rather than some external or abstract references.
- Perceived wholes are considered during part searches rather than predefined categories of shapes such as line, plane and so on. Similarly, any distinction between line shapes and curved shapes disappeared in terms of part embeddings.
- The method also introduces emergence based on designer defined constraints, like embedding wholes as parts although some parts of the initial shape are missing.

In addition to the proposed solutions to the part embedding problem, a novel critical point detection method is developed as a supplementary method to determine the internal constraints within shapes. The motivation of this method was to find a robust method to automatically extract critical points within complex line drawings - like Seljuk patterns - that can serve as registration marks during part searches. This aim is achieved by the development of the novel iterative method which successfully identifies a set of critical points after a two-phased analysis.

1.4 Thesis Overview

In this chapter, the reader is provided the necessary background information related with the shape algebras, part and embedding relations and the current state of the literature. The methods that are original to this thesis are discussed in Chapters 2, 3 and 4.

The solution for the part detection problem for line shapes is introduced in Chapter 2. The work offers a simple and practically available algorithm to the problem utilizing two different algebraic levels. A mapping to the lower level algebra introduces the set of registration points that are utilized to constrain the combinatorial searches during part embeddings. Representation of shapes is based on a novel data structure called over-complete graphs. Over-complete graphs are suitable for representing the perceived holes and operating on them effectively. The proposed solution is discussed utilizing four sample shapes each corresponding to a different case in this problem. The samples altogether represent all possible cases in the problem domain. The proposed method provides a novel solution for the nondeterministic type searches with the notion of reference shapes. All the algorithmic details of the solution is included in this chapter.

Our second approach to the part detection problem is introduced in Chapter 3. The solution is based on weighted shapes obtained from mapping line shapes to a higher level algebra. Weights define the necessary constraints for aligning two shapes in an evolutionary approach implemented as a genetic algorithm. In this solution, instead of the analytic abstractions to represent shapes, the perceptual wholes are utilized. By this way the distinction between linear or curved shapes and between line shapes and planar shapes disappeared. The additional benefits of the method are discussed by providing many visual results that belong to some experiments performed with the developed implementation in the last part of the chapter.

Chapter 4 introduces a novel method in order to locate the critical points of line drawings. The necessary background information for this work is provided in the introductory section of this Chapter. The method is based on the idea that any point that is located in a neighborhood where the mean intrinsic properties deviate significantly from that of a straight line is critical. In order to determine the deviation, any mathematical abstractions to widely accepted significance measures are avoided. Instead, a field based analysis for digital curves is performed which is true to the continuous nature of curves. In this respect, the solution provided for this method is also coherent with the solutions provided for the part detection problem. A simple to implement two-phased iterative algorithm is provided in detail. The experimental results of the method are depicted for a set of representative shapes.

The final chapter concludes the work discussing the possible future directions of this research.

CHAPTER 2

EMBEDDING SHAPES WITHOUT PREDEFINED PARTS

Embedding a shape within another is a challenging problem in digital computers especially when shapes should be treated continuous as how they are in shape grammars. This is achieved by allowing different interpretations of shapes via their projections to a lower algebraic level. Continuity in the representation is achieved with the over-complete character of the proposed representation, which depends on *perceptual wholes*[1] obtained from the interpretations of these projections, rather than some geometric notions or symbolic interpretations. This representation provides necessary tools to make part embeddings true to the continuous nature of the shapes and to the shape grammar formalism.

The chapter begins with an introduction to the proposed shape representation which is suitable for part embeddings. In this representation, two different algebraic levels of the shape grammar formalism is utilized. The analysis in the two algebraic levels is achieved via the projections of the shapes from one level to the other. Each projection identifies different parts of a shape by means of their boundaries. Hence, each such projection is actually a different decomposition of a shape, serving for a temporary representation. In these representations, the boundary elements that define the parts of a shape, hence the set of constraints within the shape, are not abstract or external to the user. Their definitions within the initial shape can change as visual rules are applied or as the process which defines the constraints change.

Following the details of the representation, the technical framework which implements the part embedding problem is provided in detail. The method which is easy to grasp and implement, interprets part relations true to visual thinking.

---

[1] Perceptual wholes are discussed in Section 2.3

2.1 Shape Representation

The most crucial part of the part embedding problem is determining a good representation of shapes, yet it determines the range and affect of the operations on shapes. The proposed representation respects the continuous nature of the shapes and is appropriate for any type of part searches[2]. This is made possible by considering shapes as a composition of parts temporarily defined by a set of boundary points. Part boundaries are exploited to define a set of possibly overlapping perceptual wholes considering the intrinsic properties of shapes.

A process, whether automated or defined according to the user/designer's selections, determines the part boundaries in a consistent manner. Accordingly, the proposed representation decomposes shapes and redefines the relations among the parts that assemble to form them. The important property of this representation is that nothing new is added to or deleted from the original shape.

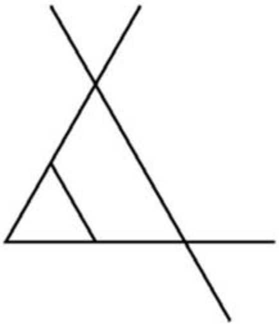


Figure 2.1: A sample line shape.

Let us explain the core idea related with the representation using a sample shape depicted in Figure 2.1. Assume that a process extracts some topological descriptions that incorporate some parts and their boundaries to define the shape[3]. According to this description, a set of points as the boundaries of shape parts are highlighted. Let these points consistently be either the intersection points or the boundary points of the maximal lines[4] of a shape .

---

[2] The nondeterministic type embeddings are included with a novel mechanism that is made clear in Section 2.6.

[3] Each such description is a set of assumed primary features and how they are assembled which is arbitrary, temporary and disputable.

[4] Refer to Chapter 1 for the definition of maximal element representation.

According to the defined process, the intersection points and the boundary points of the four maximal lines in Figure 2.1, are identified as the critical points (Figure 2.2). Then, some line segments are implicitly pointed out as being bounded by these points. It is wise to restate here that these points are selected by the defined automated process which considers a perceived topology to illustrate the solution for a generic case. They can be assigned in other cases even manually, in many different ways depending on what is intended with the choice of the set of critical points. Since, part definitions are not regularized in this approach, the regularity should be imposed on the part boundaries in order to assure the consistency between the key features of the two shapes that are utilized during matching.

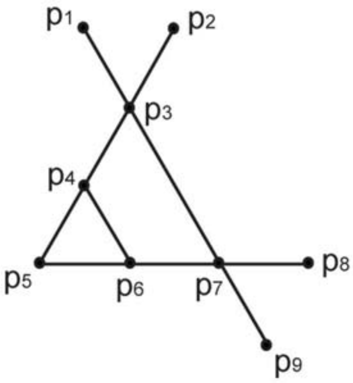


Figure 2.2: Topologically critical points are labeled. Apart from the two intersection points $\{p_3$ and $p_7\}$, all the remaining points are the boundary points of the maximal lines.

All these critical points are utilized to analyze a given shape in a different algebraic level to extract the relations about the embedded parts of a shape. For the given example shape, which is in U_{12}, the projection of the shape to one step lower level, to U_{02}, and attributing that shape with labels so as to encode some relations, in V_{02}, results in an efficient representation for embedding parts[5].

2.1.1 Mapping U_{12} to V_{02}

Let us define a shape in U_{12} with non-overlapping line segments which are bounded by a set of critical points as:

---

[5] For the generic case a shape can be considered in U_{00} algebra and be projected into V_{02}

$S_{U_{12}} = \{l_1, l_2, ..., l_n\}$ where, l_i is a line segment.

In this case, the definition of the critical points of a shape can be unified to a class of boundary points of the said line segments instead of the two classes we defined previously as the intersection and boundary points. 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:

$M : U_{12} \rightarrow U_{02}$
$M(S_{U12}) = \{b(l_1), b(l_2), ...b(l_n)\}$

where *b* is the boundary operator:

$b(l_i) = \{p_{i_1}, p_{i_2}\}$, where p_{i_1} and p_{i_2} are the boundary points of the line segment l_i.

This operator generates a mapping of a 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.

$M : U_{12} \rightarrow V_{02}$
$M(S_{U_{12}}) = \{(p_{11}, L(p_{11})), (p_{12}, L(p_{12})), (p_{21}, L(p_{21})), ..., (p_{n_1}, L(p_{n_1})), (p_{n_2}, L(p_{n_2}))\}$, where $L(p)$ is the list of labels for point p.

The labels encode the relations between the points in U$_{02}$ and refer to the line segments in U$_{12}$. As a result, for each point, a set of line segments that the point is coincident with is stored in a list of labels.

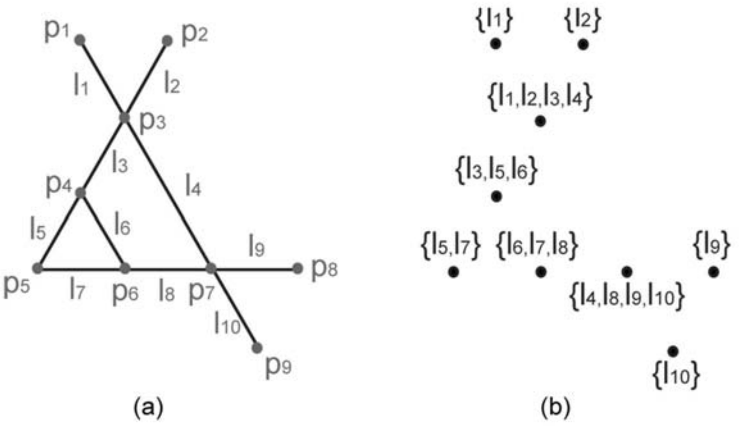


Figure 2.3: (a)Shape in $U_{12}.V_{02}$ with labels for line segments and points shown for illustrative purpose, (b)The corresponding shape in V_{02} obtained with a particular mapping.

The shape obtained after this mapping contains a finite number of points and associated labels (Figure 2.3)-(b). Labels encode the information of how a point connects to the other points. Labels can be changed, altering the connection information for alternative readings of the shape in V_{02} without altering the shape. A shape can also be represented with different V_{02} mappings as dictated by the selection of the critical points. All these V_{02} representations can be added together to get a new V_{02} representation, all without altering the shape.

2.1.2 Attributed Undirected Graph Representation

In the proposed solution, shapes are represented as attributed undirected graphs. Attributed undirected graphs are very suitable data structures to represent V_{02} shapes which are computed, in the context of this work, with a particular mapping M as the projections of the U_{12} shapes (Section 2.1.1). The nodes (vertices) of the graphs refer to the points in V_{02} and the edges which code the connection relations between the nodes are inferred from the labels. Graph node attributes are extracted from the line segments (in U_{12}), which are inferred from the labels (in V_{02}). From here on, attributed undirected graphs will be referred as *shape graphs*.

Formally, a shape in V_{02} is represented as a shape 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:

- if $L(n_i) \cap L(n_j) \neq \emptyset$ then $(n_i, n_j) \in E$. ($L(n)$: set of labels associated with node n).

Graph attributes should be set in a way to keep the necessary information for the efficient computations with shapes. In the proposed representation, graph attributes are set for each graph nodes to store some information about the line segments corresponding to the labels and the related boundary points. For each node n_i, the attributes are :

1. Node position: Local image space coordinates of the corresponding boundary point,
2. Edge information: Necessary information about the line segments that a node has a relation (defined in V_{02}).

In the sample implementation that we explain in the Section 2.7 briefly, the edge information attributes of a node n_i is set for each connected node (n_j) of (n_i) as:

(a) Node-to-node connection angle: The angle with the positive x-direction when a line drawn from node n_i to the node n_j. (Figure 2.4)

(b) Node-to-node connecting line segment: The line segment in U_{12} that is bounded by the nodes n_i and n_j.

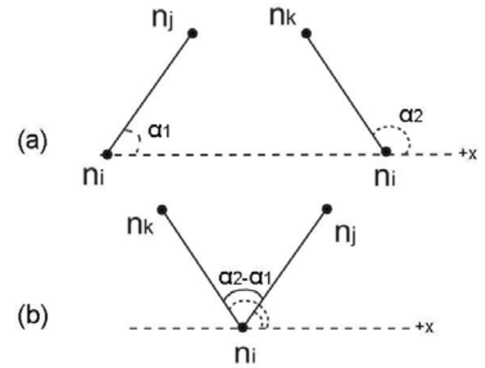


Figure 2.4: Sample node-to-node connection angles between the nodes (n_i,n_j) and (n_i,n_k): α_1 and α_2. (b) The edge connection angles between the edges connecting (n_i,n_k) and (n_i,n_j): ($\alpha_2 - \alpha_1$). It is computed via node-to-node connection angles when necessary.

Let us refer to the shapes that are composed of a finite number of straight lines, such as the one in Figure 2.1, as linear shapes from here on. For efficiency reasons, if it is known that a given shape is a linear shape, it is sufficient to set a flag to imply this property instead of keeping node-to-node connecting line segments as the graph attribute that we mentioned above, in the item (b). For a more general case, like the one shown in Figure 2.5, where no presumption could be made about a given shape, the line segments within the shape should be kept as the graph attributes to validate part embeddings.

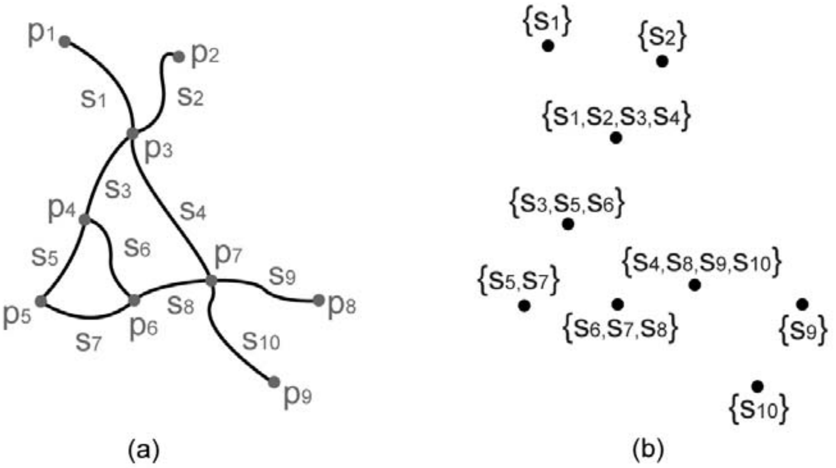


Figure 2.5: Sample shape for a generalized case. Linear or not, segments of U_{12} elements, have a boundary elements set in V_{02}.

2.2 Shape Graphs

2.2.1 Equivalence Relations

A shape can occur in different instances in a digital medium where its scale, orientation and position change (Figure 2.6). Although the perceived images for each instances are different, the shapes in all these instances are equivalent under some Euclidean transformations. In this section, we develop the equivalence relation between two shape graphs that are shape equivalent under some Euclidean transformations.

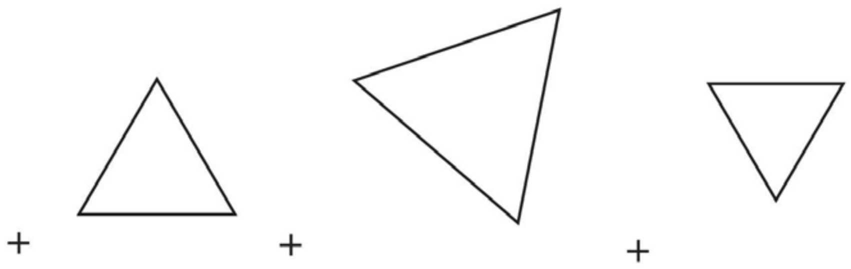


Figure 2.6: Three different instances that are shape equivalent under Euclidean transformations.

The mapping function, M, described in Section 2.1.1 is not a one-to-one relation when shapes are considered. Each of the different instances of a shape is mapped to a different graph. Therefore, two different graphs may actually refer to the same shape.

The graph representations show the assumed topologically critical points and relations between these points. For a given shape, the relations between the nodes in the graph representations do not change when shapes go under Euclidean transformations. Only, the attributes of the graphs, namely absolute positions of the points and scales and orientations of node-to-node connecting lines may change. For seeking an equivalence relation between two linear shapes (like the shapes in Figure 2.6), 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,

$f : N(G_1) \rightarrow N(G_2)$

such that any two nodes n_i and n_j of G_1 are adjacent in G_1 if and only if $f(n_i)$ and $f(n_j)$ 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. Edge connection angle between $< n_j, n_i, n_k >$ nodes of G_1 centered at n_i and corresponding to angle centered at $f(n_i)$ in G_2, $< f(n_j), f(n_i), f(n_k) >$ should be equal;
2. The edge length ratios:

   $r_{G_1} = \frac{||P(n_j)-P(n_i)||}{||P(n_k)-P(n_i)||}$, (where $0 < r_{G_1} \leq 1$) of G_1

   and

$r_{G_2} = \frac{\|P(f(n_j))-P(f(n_i))\|}{\|P(f(n_k))-P(f(n_i))\|}$, (where $0 < r_{G_2} \leq 1$) of G_2

should be equivalent. Note that $P(n)$ is the local coordinates of a node n.

For a generalized version of the equivalence relation[6], an additional constraint should be satisfied to validate the equivalence relations of two shape graphs, which is stated below.

3. In addition to validating the constrained structural isomorphism, all pairs of node-to-node connecting line segments in two graphs should be equivalent.

The shape equivalence 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 condition but not sufficient for being in the same shape equivalence class. The two graphs should also satisfy the constraints defined in (1), (2) and (3).

2.2.2 Part Relations

Let us assume that we are to detect shape (a) as a part of shape (b) in Figure 2.7.

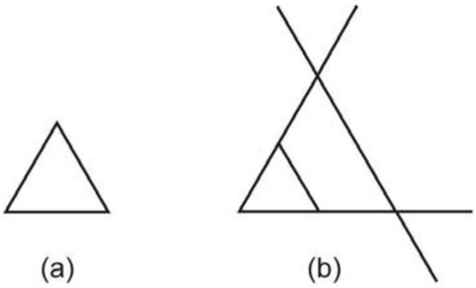

(a) (b)

Figure 2.7: Sample shape (a) to be detected in shape (b)

There are more than one occurrences of (a) within (b) in different scales and orientations. When the shape graph of (b) is traversed to obtain the sub-shapes which are shape equivalent

---

[6] What is meant by a generalized version of identity relation is an identity relation that applies to any kind of shapes; not limited to the shapes with straight lines.

to (a), there are two simple[7] subgraphs for which nodes and node connections are shown in Figure 2.8.

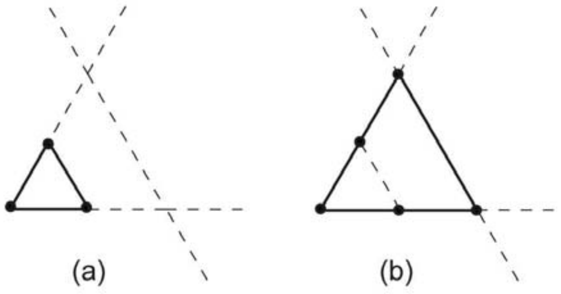


Figure 2.8: Two subgraphs for the shape graph of the Figure 2.7

Although both (Figure 2.8-(a) and (b)) are shape equivalent to Figure 2.7-(a) under some Euclidean transformations, Figure 2.8-(b) is not graph equivalent to Figure 2.7-(a) as it has more nodes than the boundaries of its maximal lines.

Instead, the corresponding subgraph representation of Figure 2.7 (b) should be as in Figure 2.9 where the only nodes that define the subgraph are topologically critical points.

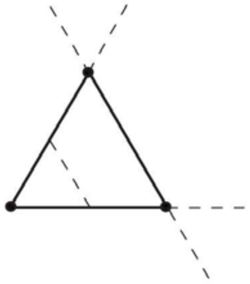


Figure 2.9: A subgraph of the graph of the shape in Figure 2.7-(b)

As it is exemplified here, such cases may usually be the case when all possible parts of a given shape is searched for, which is the problem that we are pursuing in this thesis. To overcome this problem, where the graph representation for the embedded part is not com-

---

[7] A simple shape graph is any ordinary attributed undirected shape graph that we have mentioned so far in this document. This tag is used here to differentiate it from over-complete graphs (Section 2.3)

pletely a subgraph of the graph representation of the shape it is a part of, the definition of simple graph is extended to obtain a new graph representation for shapes that we refer to as over-complete graph representation. In an over-complete graph, 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.

2.3 Over-Complete Graph Representation

As seen in Figure 2.8-(b), the criticality of a point depends on the context. If two edges connecting through a node have similar attributes as co-linearity or continuity, we can choose not to perceive the point at the node as a critical point and perceive the concatanation of two edges as a whole, calling it a *perceptual whole*. Each perceptual whole creates a new connection between the two previously unconnected nodes that are located at the other ends of these two edges and leads to an alternative traversal of the simple shape graph. In other words, we perceive the points in U_{02} and V_{02} in an alternative way, without altering the U_{12} shape. Over-complete graph representation is an extension to the simple shape graph representation such that in addition to the simple node connections, all the nodes that are at the boundaries of the perceptual wholes have connections. By means of this extension all implicit connections are made explicit, and the subgraph shown in Figure 2.9 become a valid subgraph of the graph in Figure 2.7-(b).

Perceptual wholes could be defined in different ways. Any particular definition of the perceptual wholes will not change the method we define here for extracting over-complete graph representations of shapes. The crucial thing is being consistent in the definition of the perceptual wholes within the shapes that are the subjects of a visual computation. In our particular implementation, we preferred to define perceptual wholes from the segments that have the first order parametric continuity (C^1). In C^1 continuity, the tangents of the segments at the vicinity of their connecting points are equal.

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 the line, boundary points of which are coincident with these nodes, forms a perceptual whole. In the case of linear shapes, we assumed that the line segment between each pair of points which are coincident with the same straight line forms a perceptual whole. Hence, there is a connection between each pair of such points. For these cases, connection angles between nodes in a simple graph facilitate the identification of a set of nodes which are coincident with the same straight line (Figure 2.10). In an over-complete graph representation, all of the nodes in this set are shown to have a direct connection with each other.

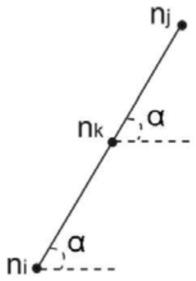


Figure 2.10: The implicit connection between the nodes n_i and n_j are made explicit in the over-complete representation of this shape.

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.

Let us explain the V_{02} interpretation of a shape in an over-complete graph representation for a more general case. Assume that a part of a shape is given as in Figure 2.11-(a) with the line fragments connecting through the critical points are labelled as shown. Mapping of the shape to V_{02} domain is as shown in Figure 2.11-(b). This is the simple mapping that we utilized during constructing the simple shape graph of the shape.

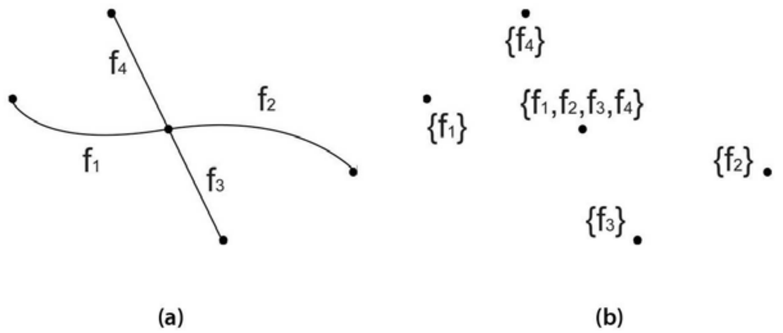


Figure 2.11: (a) Sample shape with labelled fragments. (b) Corresponding V_{02} shape

The two line fragments f_1 and f_2 meet at the center node with a smooth and continuous connection (satisfying the C^1 continuity) (Figure 2.12-(a)). Therefore, we can choose not to see the center node and perceive the concatanation of f_1 and f_2 as a whole ($f_1 f_2$). In the V_{02} shape, which will eventually be used to construct the over-complete graph, the label corresponding to this whole is associated with the boundary points of the whole (Figure 2.12-(b)).

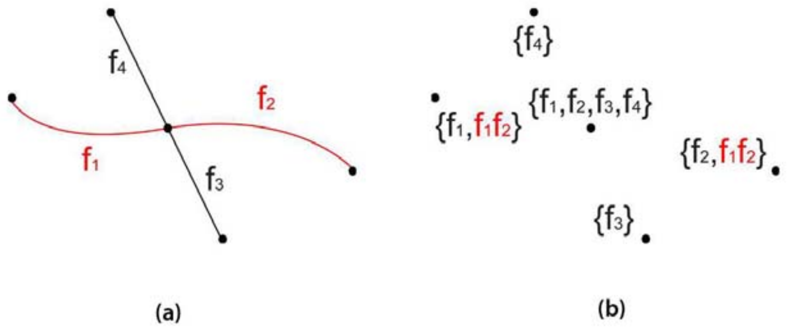


Figure 2.12: (a) Concatanation of f_1 and f_2 forms a perceptual whole. (b) Node relations through $f_1 f_2$ segment is reflected to V_{02} shape.

Similarly, concatenation of the two line fragments f_3 and f_4 can be perceived as a whole when we choose not to see the center node. In that case, the label ($f_4 f_3$) corresponding to the concatanated lines is assoicated with the nodes that are coincident with the boundary points of that whole (Figure 2.13).

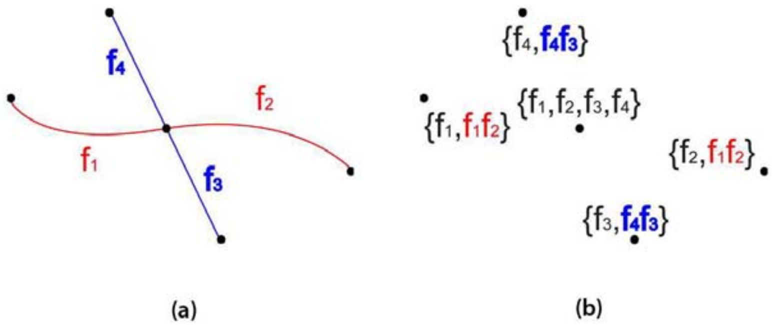


Figure 2.13: (a) Concatanation of f_4 and f_3 forms a perceptual whole. (b) Node relations through $f_4 f_3$ segment is reflected to V_{02} shape.

The V_{02} shape in Figure 2.10-(b) is mapped to a simple graph as shown in Figure 2.14-(a). On the other hand, the new interpretation of the same shape which considers the perceptual wholes introduces new connections when V_{02} shape shown in Figure 2.13-(b) is mapped to a graph (Figure 2.14-(b)).

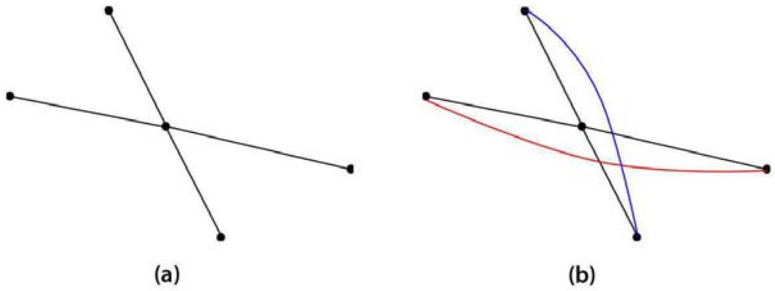


Figure 2.14: (a) Simple shape graph of the shape in Figure 2.11-(a). (b) Over-complete graph of the same shape

Accordingly, the simple graph and the over-complete graph of the curvilinear shape in Figure 2.5 are depicted in Figure 2.15-(a) and (b) respectively. Notice the additional connection relations in the over-complete representation. These connections help extending the search space when part relation is asked for two shapes. This extended structure makes graphs suitable to locate every possible (if required) occurrence of a given shape within another.

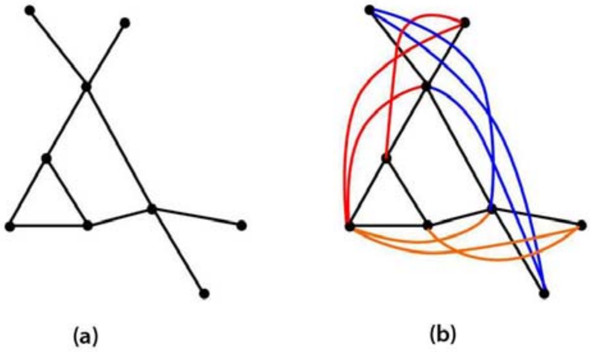


Figure 2.15: (a) Simple graph of the shape in Figure 2.5. (b) Over-complete graph of the same shape.

2.4 Embedding Parts

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 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.

Before going into the details of the sub-shape embedding algorithm, let us make some definitions related with the node types, which will guide the explanation of the proposed algorithm in the next section. Nodes in a given source graph can be categorized as free or non-free. A node which is connected to more than one node is a non-free node, whereas a node which is connected to just one other node can be a free node or non-free node according to the constraints defined externally for that node. When no external constraints are defined, a node with just one connection is treated as a free node. Free nodes are never located at an intersection point while non-free nodes are. Therefore, non-free nodes play an important role in finding the corresponding nodes of two shape graphs in the problem of searching for parts.

Sub-shape embedding 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 candidate 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 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 2.16). For this reason, in the second step, the shape algebra extends to include the line segments in $U_{12}.V_{02}$.

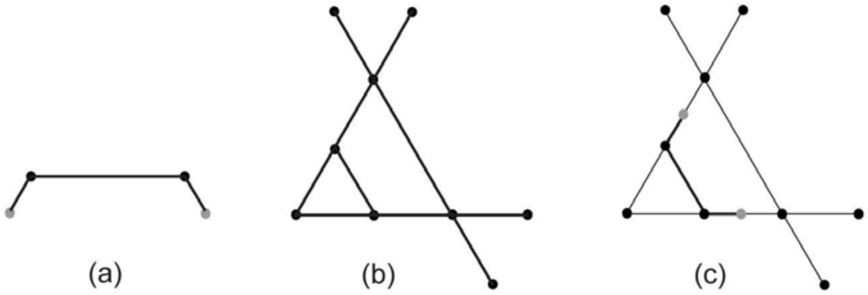


Figure 2.16: Source graph: free node points are shown in grey color, (b) Target graph, (c) Free nodes are not coincident with any node of the target graph.

If the source shape is a 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 sub part 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 2.16-(c).

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 as 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 source shape for each possible mapping.

2.5 The Two Step Part Embedding Algorithm

In this section, a two step part embedding algorithm is provided referring to the four exemplary cases that are illustrated in Figure 2.17. Each case is a sample for a different problem in part embedding when their representative graphs are considered. Altogether they represent all the possible cases, hence the proposed algorithm provides solutions for each of them. Note that the Second Pass of the algorithm provided in this section is an optimized version of a general solution to make it more efficient for the linear shapes like the selected exemplary cases. Extension of this algorithm for a generalized solution, which additionally requires matching any kind of line segments (i.e. curves, arcs, etc.) in the Second Pass, is discussed in Section 2.7.

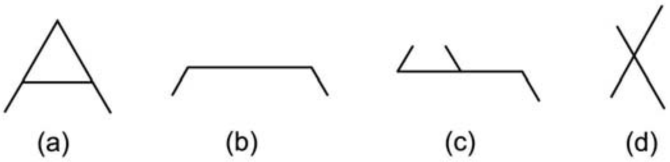


Figure 2.17: Four exemplary cases to discuss the part embedding algorithm

The initial shape selected for the exemplary cases is already given in Figure 2.1.

2.5.1 First Pass

1. Construct the source graph $G_S = (N_S, E_S)$ and over-complete target graph $G_T = (N_T, E_T)$.

2. Enumerate the nodes in the target graph $(n_1{}^T, n_2{}^T, ..., n_k{}^T)$, where k is the total number of nodes in the over-complete target graph.

3. From the source graph, select a subset of the non-free (NF) source nodes in order to compute the candidate transformations that transform the source graph nodes into the target graph domain to detect embedded parts. There are four different cases for selecting this subset regarding the number of NF nodes in the source graph.

    (a) **Case 1**: *There are three or more NF nodes in the source graph such that at least* *three of them are non-collinear (Figure 2.17-(a) and 2.18)*

i. Select three NF source nodes $S_{NF3} = \{n_1{}^S, n_2{}^S, n_3{}^S\}$ and construct the corresponding 3-node vertex-induced subgraph of the source graph $G_S[S_{NF3}] = (N_{S3}, E_{S3})$.

ii. Create a set of candidate node mappings π by generating all three-node permutations of the target nodes. There are $k.(k-1).(k-2)$ number of permutations. Four sample mappings are shown in the first row of Figure 2.18.

iii. Construct 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 $1 \leq i, j, m \leq k$.

iv. 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 2.4) among the source nodes in N_{S3} must be compatible with the ones among the mapped target nodes in N_{T3}.

    D. List every mapping which satisfies these three conditions.

v. For each compatible mapping in the list, compute the transformation matrix Γ. Since the source node coordinates and the corresponding target node coordinates are known for three non-linear points, the transformation matrix can be computed uniquely.

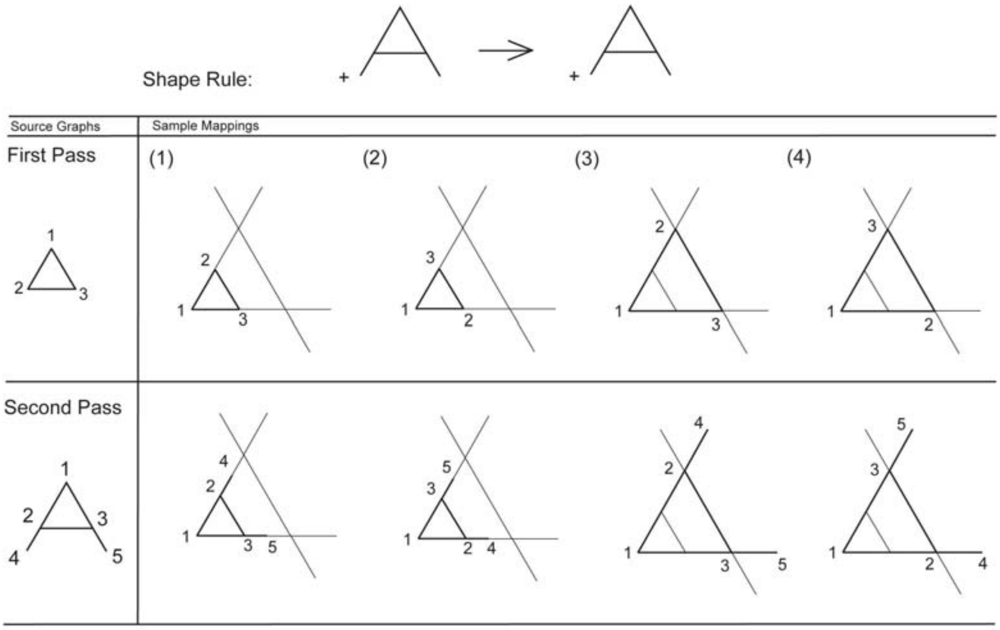


Figure 2.18: Example rule for *Case 1* is displayed for four sample mappings (1 – 4). The candidate mappings are displayed in the first row. A 3-node source subgraph is selected from the source graph and shown on the left column of the first row. In the second pass, illustrated in the second row, the 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 indices.

(b) **Case 2**: *There are exactly two NF nodes in the source graph (Figure 2.17-(b) and Figure 2.19)*

  i. Select two NF source nodes $S_{NF2} = \{n_1{}^S, n_2{}^S\}$ and construct the corresponding 2-node vertex-induced subgraph of the source graph $G_S[S_{NF2}] = (N_{S2}, E_{S2})$. (Selected NF source graph nodes are depicted with the indexes 1 and 2 in the left column of the first row in Figure 2.19.)

  ii. 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 column of the first row in Figure 2.19.)

  iii. Construct two-node vertex-induced subgraph of the target graph for each permutation $T_{N2} = \{n_i{}^T, n_j{}^T\}$ as $G_T[T_{N2}] = (N_{T2}, E_{T2})$, where $1 \leq i, j \leq k$.

  iv. There are two node mappings between the source graph and the target graph. Therefore, a unique transformation matrix can not 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 non-free source nodes, say ${n_2}^S$, perpendicular to the line connecting two non-free source nodes. The length of the external line is equal to the line connecting two non-free source nodes and the selected non-free 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:$\{{r_1}^S, {r_2}^S\}$ (as shown in the source graph of the first row in Figure 2.19). Similarly, an external line is drawn which passes through the target node corresponding to the selected non-free source node, say ${n_j}^T$, 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 $({r_1}^T, {r_2}^T)$.

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

vi. For each mapping μ_1 and μ_2, compute a unique transformation matrix Γ.

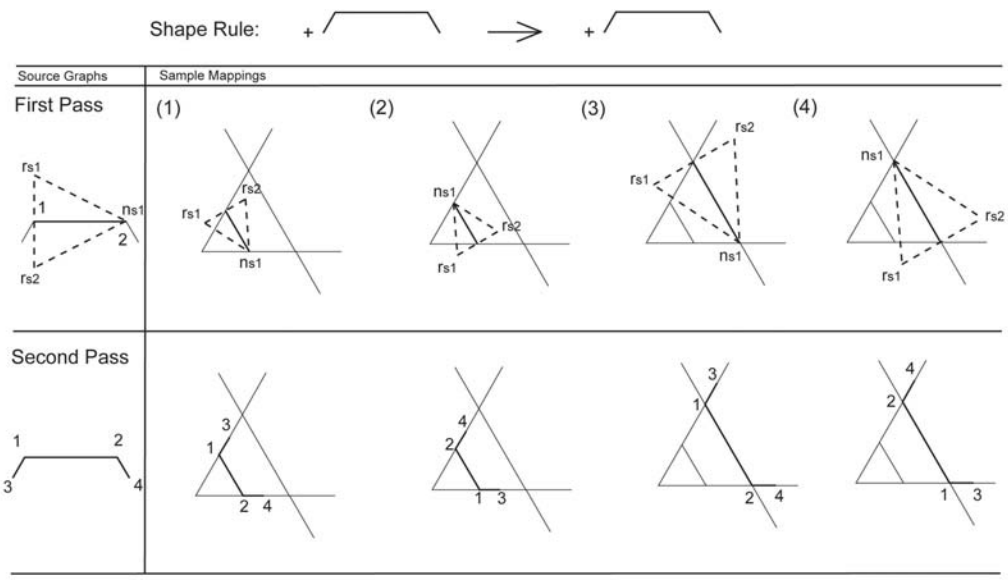


Figure 2.19: Example rule for *Case 2* is displayed for four sample mappings in (1-4). The graph displayed with dashed lines in the first row contains nodes that are constructed with external reference points explained in *Case 2-4*. In the figure above, the source graph nodes ${n_1}^S, {r_1}^S, {r_2}^S$ are labeled as n_{s1}, r_{s1}, r_{s2} respectively.

(c) **Case 3**: *There are more than two NF nodes in the source graph all of which are collinear (coincident with the same line) (Figure 2.17-(c) and Figure 2.20):*

    i. Select 2 NF source nodes randomly and compute the transformation matrices following the algorithmic steps described for Case 2.

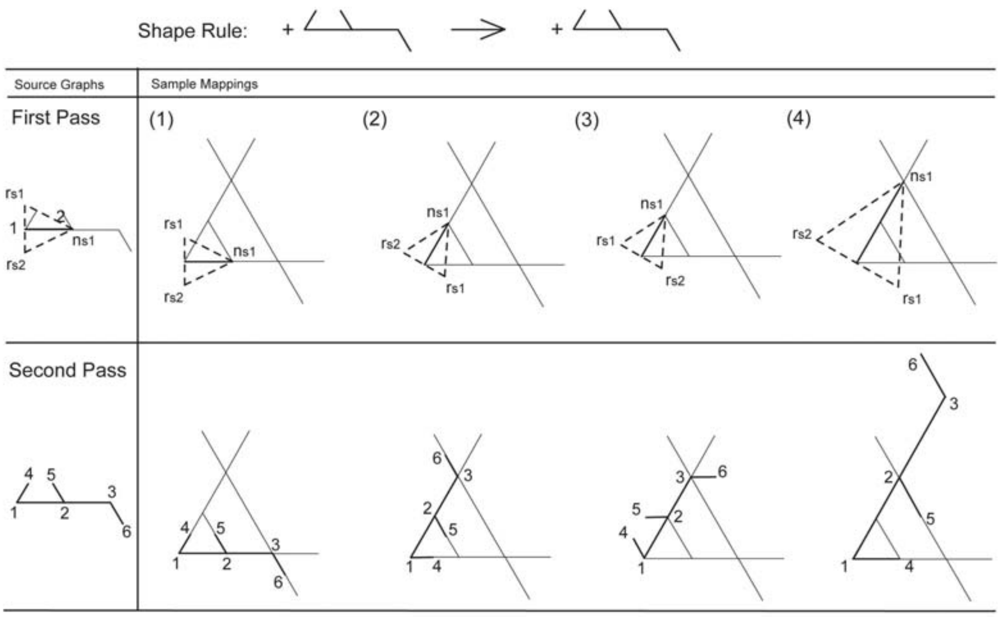


Figure 2.20: Example rule for Case 3 is displayed for four sample mappings in (1-4). The mappings in (1) and (2) are acceptable mappings for which the source shape is a part of the target shape, while (3) and (4) are rejected mappings in the second pass of the algorithm. In the figure above, the source graph nodes ${n_1}^S, {r_1}^S, {r_2}^S$ are labeled as n_{S1}, r_{S1}, r_{S2} respectively.

(d) **Case 4**: *There are less than two NF nodes in the source graph (Figure 2.17-(d) and Figure 2.21):*

   i. If a reference shape is provided (see the discussion related with the reference shapes in Section 2.6):

      A. Compute the reference shape graph $G_R = (N_R, E_R)$.

      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 type non-free by the definition of the reference graphs.

   ii. If there is no reference shape, Assume that all nodes of the source shape is of type non-free. Compute the transformation matrices as in *Case 1* or *Case 2* depending on the number of source nodes.

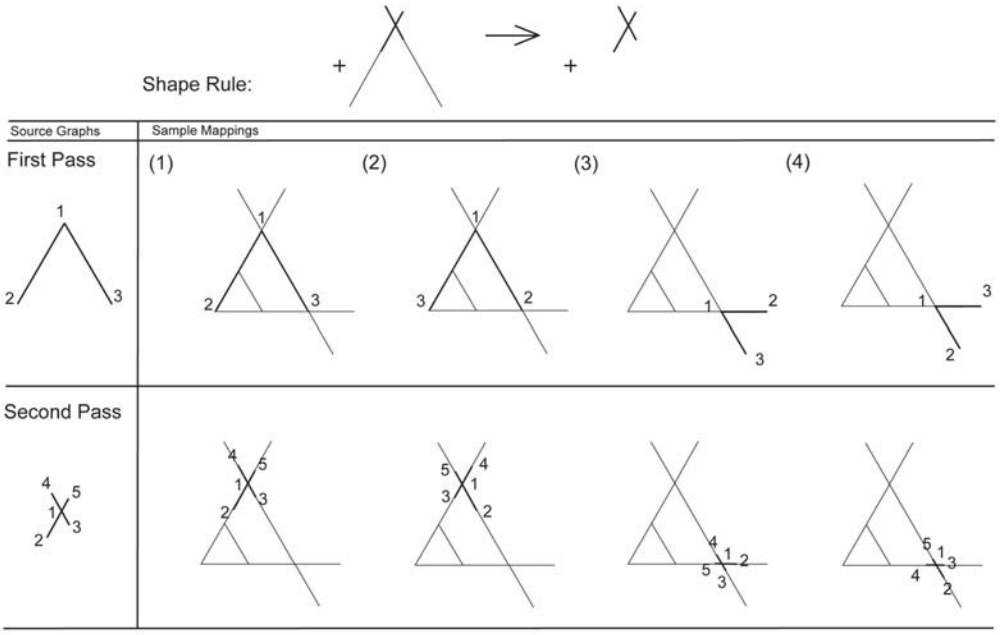


Figure 2.21: Example rule for *Case 4* is displayed for four sample mappings in (1-4). A user defined reference shape is introduced along with the left hand side of the rule, which is displayed with light-weighted lines in the rule definition. Since all the nodes in a reference shape graph is of type NF, 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.

2.5.2 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 non-free nodes) of the source graph using the transformation matrix (Γ) computed in the first step.
    (b) Compute the node correspondences, μ for all the transformed non-free source nodes with the target graph nodes comparing their coordinates. If there is any non-free source node which does not correspond to any target node, discard this mapping from the candidate mapping list, e.g. in Figure 2.20, NF node indexed as 3 in the sample mapping (4).
    (c) Compute the node correspondences, μ for all free source nodes:
        i. If transformed free source node is coincident with a node of the target graph, associate the target node index with the corresponding free source node, i.e. in Figure 2.20, free node indexed as 4 in the sample mapping (4).

        ii. If 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, i.e.: in Figure 2.21, 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, i.e. in Figure 2.20, nodes indexed as 4, 5 and 6 in sample mapping (3).

2. Check 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 $(n_i^S, n_j^S) \in E_S$, whether there exists a relation $(\mu(n_i^S, n_j^S)) \in E_T$ or not. If there exists any $(n_i^S, n_j^S) \in E_S$ such that $(\mu(n_i^S, n_j^S)) \notin E_T$, this mapping is deleted from the candidate mapping list.

    (b) If for some of the free source nodes, there are more than one associated target nodes (see 1.c.ii). Assume that $(n_i^S, n_j^S) \in E_S$ and $\mu(n_j^S) = \{n_a^T, n_b^T, \}$.

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

        ii. If $n_a^T = \mu(n_i^S)$ and $n_b^T \neq \mu(n_i^S)$, then there must be $(\mu(n_i^S), n_b^T) \in E_T$. Otherwise, discard this mapping.

        iii. $n_b^T = \mu(n_i^S)$ and $n_a^T \neq \mu(n_i^S)$, then there must be $(\mu(n_i^S), n_a^T) \in E_T$. Otherwise, discard this mapping.

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

2.6 The Reference Shape

According to the proposed part embedding algorithm (Section 2.5), if the source graph contains less than two non-free nodes, there will be infinitely many transformation possibilities.

We refer to the searches with this type of source graphs as the nondeterministic type searches, since embedding the part in a unique way is not possible. Producing all possible solutions is not feasible both for time and storage limitations of the computing medium. Hence, a novel visual communication system is introduced in this section which defines necessary constraints for mapping a given source shape to a part of a given target shape based on user(/designer)'s intentions.

In this system, the communication between the computing machine and the user is achieved by means of a visual rule definition in U_{12}. However, the left hand side of the rule is defined in a particular way (Figure 2.21). The original left hand side of the rule is defined in relation with another shape, called *the reference shape*, which is labeled differently, i.e. lies in a different layer. During the operations, the user defined reference shape is mapped to 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 graph node types are modified to be of *non-free typed* even for the free nodes of the original reference graph. The reference shape, as its name implies, acts like a reference to compute the candidate transformations in the fist place. These candidate transformations are directly utilized to generate the set of successful embeddings of the source shape according to the relations between the source shape and the reference shape.

There is no predefined restriction on the definition of a reference shape. It can be defined in any way that is meaningful for a user. For example, in Figure 2.22, a change in the scale of the reference shape without changing the visual rule applied in Figure 2.21 significantly changes the parts detected in the initial shape.

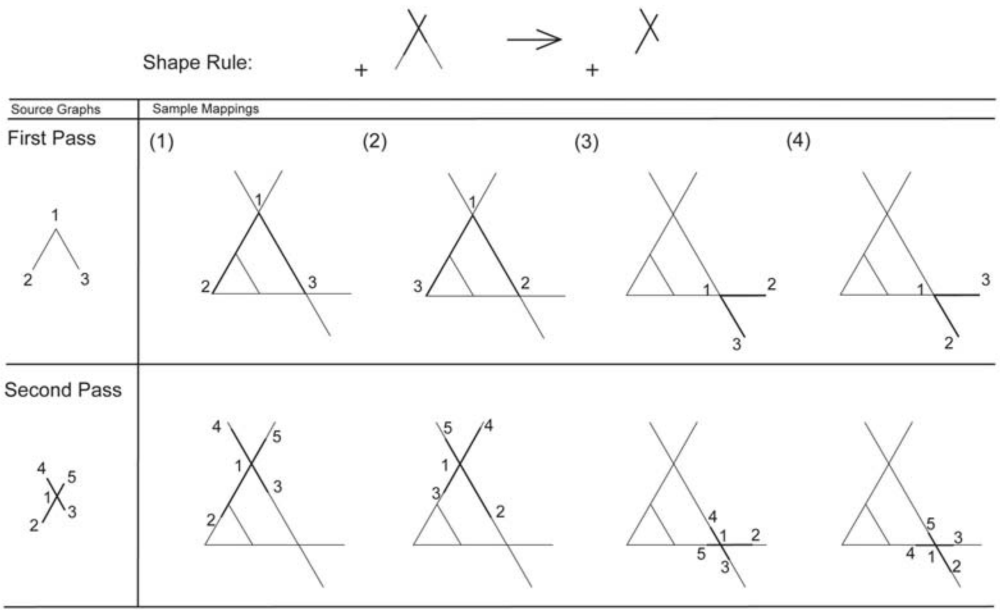


Figure 2.22: Application of the same rule in Figure 2.21 with a different reference shape.

The solution proposed for this type of searches enables the user for embedding any source shape to a target shape by specifying a proper reference shape (Figure 2.23).

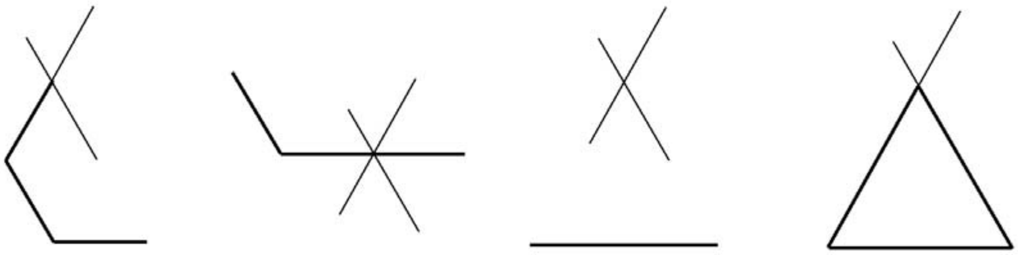


Figure 2.23: Reference shapes can be defined in any desired way. Some sample reference shapes for the same rule in Figure 2.21 are displayed with bolder lines.

The notion of the reference shape relaxes the predefined constraints for embedding nondeterministic type shapes. Such a fixed solution would contradict with the philosophy of visual computation.

2.7 Sample Implementation

In order to complement the solution we provided for the part embedding problem, the proposed algorithm is integrated into a sample shape grammar interpreter implementation. Considering that there is no practically accessible application in the literature that solved part embedding problem supporting any type of line shapes, releasing a working application with the proposed structures and the algorithm has a practical value. Not concentrating on all the implementation details, here only certain technical aspects of the approach will be discussed.

The implementation is developed in MATLAB (7.0) environment and supported with a graphical user interface (GUI) via which the shape rules can be introduced to the system directly with image files. When an image file is loaded through the GUI, the following computations are performed in that order to construct the proposed representation:

1. Extract a set of critical points (CP) to transform the given shape to a U_{02} shape (Chapter 4). The set of critical points partitions the original shape into a set of line segments (fragments) which are connected to some other through these points to form the original shape.
2. Find the relations among CP set and the set of line segments to construct the V_{02} shape.
3. Construct the corresponding shape graph.

After the construction of the simple shape graphs, over-complete graph representation is constructed by evaluating the connections in the vicinity of the nodes to determine possible perceptual wholes. For the linear cases, over-complete graph is constructed only for the target (initial) shapes, while the source shapes are represented with simple graphs. The reason can be explained as follows. In order to embed a shape within the other a set of points are utilized as registration points while determining possible correspondences. Let us refer to these points as registration marks from here on. In the proposed approach, the set of critical points extracted from shapes serve as the registration marks in embedding one shape within the other. In order this idea be effective, the registration marks should be extracted in the same way from both the source and the target shapes, in other words they should all carry the same logical meanings. For linear cases, all the registration points that correspond to the non-free nodes in the source shape graph is equally important for that shape with respect to matching. This is due to the fact that, the process which determines the critical points from the shapes (source or target shape) guarantees to mark any intersection point as the critical point. Therefore, any critical point, corresponding to a non-free node, in a simple source graph would exist in the target graph if the source shape is a part of the target shape. Hence, representation of the source shape as a simple graph is sufficient to determine three (most probably connected) non-free nodes to initiate the search in the First Pass.

On the other hand, when the shape has curved components, there are different types of critical points in addition to the intersection points. Leaving the extraction of the CPs to Chapter 4, a critical point comes with a weight that shows the strength of that point as a registration mark for a more general shape. From a set of such critical points for a source shape, the strongest three of them would serve better to find the possible transformations in the First Pass of the proposed algorithm. While selecting three strongest nodes, selecting a set of internally-connected ones would constrain the search space more than an unconnected set (due to the pruning during subgraph searches) and as a result would be more time efficient. In an over-complete graph representation, many nodes which are not immediate neighbors have direct connections due to the perceptual wholes. Hence, the probability of finding three strongly weighted points that are connected in that representation is higher than the simple graphs. For this purpose representing the source shapes as an over-complete graph is more suitable for the generalized solution.

As it is stated in Section 2.1.2, the implementation can be optimized for the linear shapes if the time efficiency in part matching is desired, yet the edge information for any linear shape is the same; straight lines. The only difference between such an implementation and a generalized solution would be in the Second Pass of the given algorithm (Section 2.5). For the linear shapes, the part relation validation for the two shape graphs would be sufficient to decide about matching, while in the generalized solution, in addition to that, node-to-node connecting line segments should also be matched for the corresponding nodes in both shapes.

In the generalized solution, Second Pass of the given algorithm is extended to include a few more steps to compare a set of line segments for the corresponding nodes of the two graphs in order to eliminate the effort for unmatching candidates beforehand. For this purpose, in the generalized GUI, all the line segment indexes that have relations with a node are kept as node-to-node connecting line segments attribute in the graph structures. There is a buffer which keeps the line segments in a canonical form as shown in (Figure 2.24) to filter out position and scale effects during comparisons.

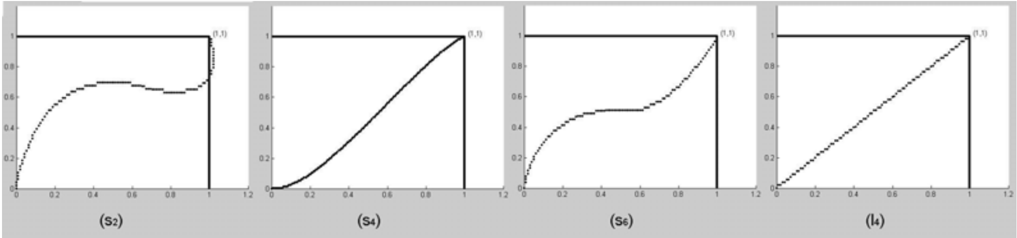


Figure 2.24: Sample Canonical representations for the curved line segments $s2$, $s4$ and $s6$ of the shape in Figure 2.5 and the straight line segment $l4$ in Figure 2.3. All canonical representations of straight line segments are similar. The only difference between two such representations may be in the density of the plotted points.

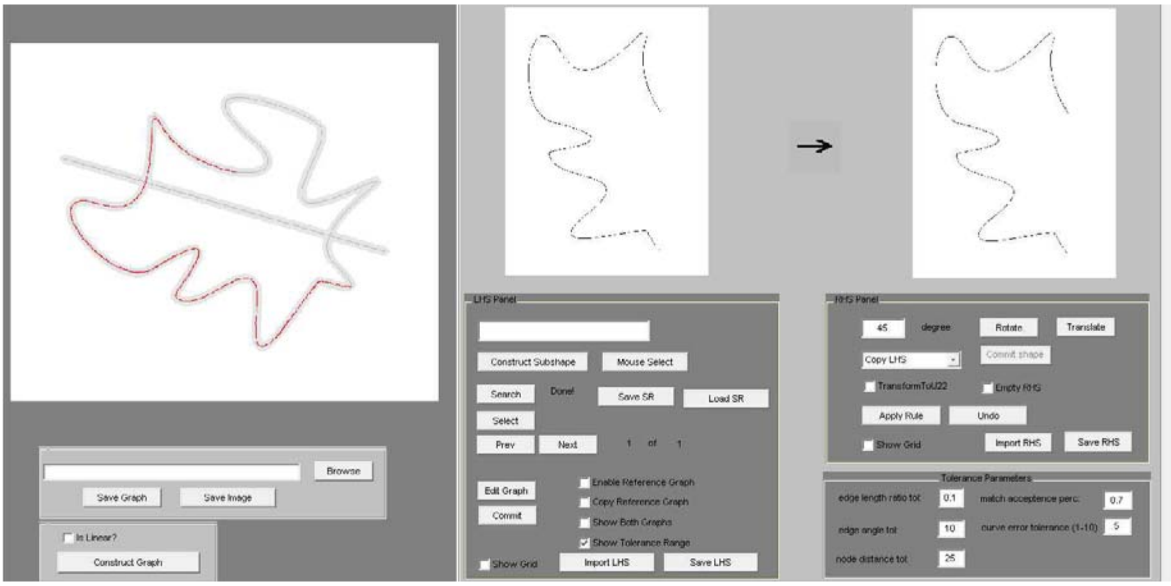
The nodal position of n_i is mapped to (0,0) and the nodal position of n_j is mapped to (1,1) in the canonical representation of a line segment bounded by n_i and n_j. The crucial point here is to make the node order compatible for the node pairs of both graphs during the matching of two graph line segments.

Accordingly, for a generalized solution, the Second Pass of the proposed algorithm is extended as follows:

1. For each possible mapping computed in the first step, find the node associations among the selected NF source nodes and the target nodes as:
    (a) Transform the selected non-free nodes of the source graph using the transformation matrix (Γ) computed in the first step.
    (b) Compute the node correspondences, μ for the transformed non-free source nodes with the target graph nodes comparing their coordinates (considering the allowed tolerance range). If there is any non-free source node which does not correspond to any target node, discard this mapping from the candidate mapping list.
2. For each selected source node ${n_i}^S$ and associated target node ${n_j}^T$:
    (a) For each line segment attribute of the node ${n_i}^S$, say ${l_{i_m}}^S$, there exists a line segment attribute, say ${l_{j_k}}^T$ associated with the target node ${n_j}^T$ in the over-complete target graph equivalent to ${l_{i_m}}^S$. If not, eliminate this mapping from the list.

3. Transform the source image with the computed transformation matrix Γ.
4. Match target image with the transformed source image by simple binary image operations considering some user defined error tolerance. if for some parts of the source image, there is no overlapping target image when the two are aligned, discard this mapping from the list.
5. Accept the remaining list of transformations as the valid set of mappings.

The GUI of the implementation, provides a generalized solution for any line shapes, like the curvilinear shape shown in Figure 2.25. Some sample part embeddings with this GUI with the generalized solution can be found in Appendix A.2.



☐ Is Linear?

Done!

1 of 1

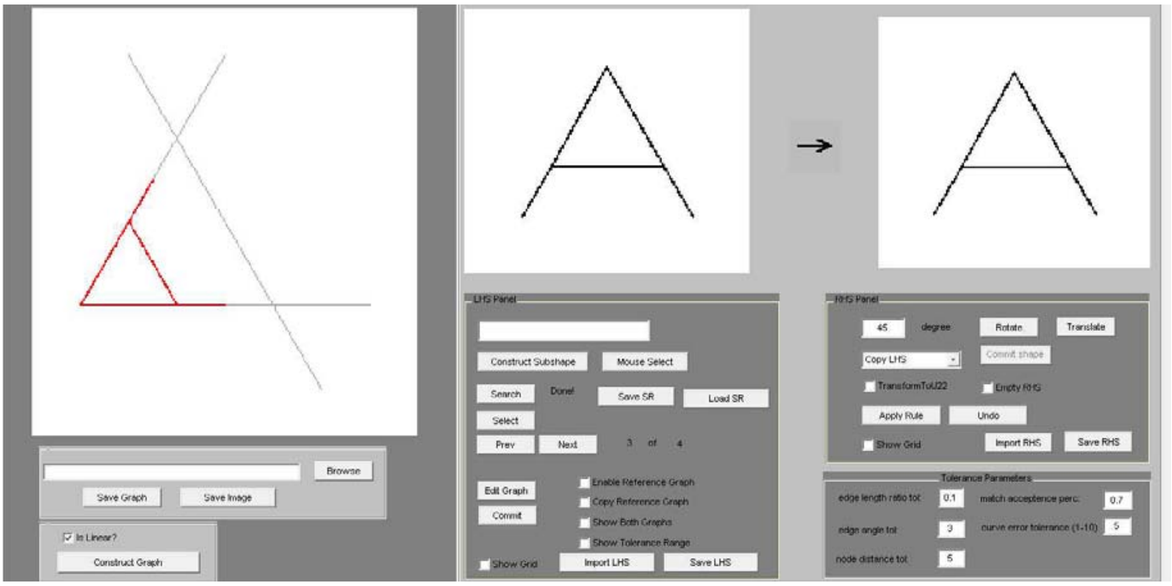
☐ Enable Reference Graph
☐ Copy Reference Graph
☐ Show Both Graphs
☑ Show Tolerance Range
☐ Show Grid

45 degree
Copy LHS
☐ TransformToU22
☐ Empty RHS
☐ Show Grid

edge length ratio tol 0.1
edge angle tol 10
node distance tol 25
match acceptence perc. 0.7
curve error tolerance (1-10) 5

Figure 2.25: Screenshot of a sample embedding within a shape with curved components.

In this implementation, part matching is optimized for linear shapes as we proposed. A sample part matching with this implementation for the example case depicted in Figure 2.17-(a), is shown in Figure 2.26.



☑ Is Linear?

Done!

3 of 4

☐ Enable Reference Graph
☐ Copy Reference Graph
☐ Show Both Graphs
☐ Show Tolerance Range
☐ Show Grid

45 degree

Copy LHS

☐ TransformToU22
☐ Empty RHS
☐ Show Grid

edge length ratio tol 0.1
match accepetence perc: 0.7
edge angle tol 3
curve error tolerance (1-10) 5
node distance tol 5

Figure 2.26: Screenshot of a sample embedding within a linear shape.

In the GUI, the alignment of the source shape with the matching part of the target shape is displayed in the left most window with the red color. The displayed sample in Figure 2.26 is one of the four possible embedding results that are detected by the application. Because of the discretization errors, we need to put some error tolerance in order to decide the node correspondences. All these parameters can be determined and changed by the user through the GUI. The results of the part embeddings with this GUI for all the exemplary cases that are shown in Figure 2.17 are provided in the Appendix A.1.

Aligning Arbitrary Line Shapes

Part embedding problem, in the context of the shape grammars, requires finding some Euclidean transformations of a shape within the other. Hence, in theory, finding a rigid transformation is sufficient to locate a part in matching. However, in the digital representation of images two different transformations of a shape may not align perfectly when one is transformed on top of the other, due to the discretization. Locating exactly the same pixels as critical points in both transformations of the same shape may not be possible all the time, although the critical point positions will be in close neighborhoods. For all these reasons, in the provided implementation, some acceptable tolerance range is defined and allowed in aligning two shapes. Considering this, in addition to the rigid transformation that align two shapes with some error tolerance, a non-rigid transformation is computed to reduce the alignment errors as much as possible.

Initially computed Euclidean transformation, which defines a rough correspondence of the two images in the allowed tolerance range, is utilized to initiate another process which computes the non-rigid transformation to find a more accurate alignment. That process uses the thin plate spline mapping algorithm to parameterize the non-rigid spatial mapping in the alignment of the two shapes. This type of non-rigid point matching requires finding a good initial alignment of the two point sets in order to map one to the other.

In the implementation, in the first phase of the method proposed in this work, a set of critical point correspondences of the two shapes is determined after the combinatorial search performed with the graph nodes. As a result, a set of candidate Euclidean transformations are determined. In the second phase, the source shape is transformed to the target shape's image domain using these transformations. Following that the alignment of the two images is checked for the acceptable tolerance range. For the set of transformations which yield the desired alignment, a rough correspondence is obtained for initiating the TPS-RPM method [15]. The true transformation which ensures the alignment of a given source shape with a given target shape is then obtained as a combination of the two transformations, namely the Euclidean transformation and the transformation obtained with the TPS-RPM method. This combined transformation should be applied to the right hand side shape when a shape grammar rule is applied.

2.8 Summary

In this chapter, a novel part embedding approach is introduced based on the temporary representations of shapes. The over-complete graph structure is suitable for this representation and for effective part searches. Graph nodes are determined according to a mapping to a lower level algebra, which is defined externally either by a user/designer or by an automated process. In that algebra points are attributed with labels to encode the relations among the shape points. During the construction of the relations, the perceptual wholes within the shapes are considered. This provides a more comprehensive way to encode the relations among the parts of the shape.

The problem space is divided into four cases and the solution for each case is discussed in detail. For the nondeterministic type searches, the notion of the reference shape is introduced a means for the communication between the user and the computing medium. This way, embedding is performed according to the user defined constraints, instead of the predefined biases.

CHAPTER 3

WEIGHTED SHAPES FOR EMBEDDING PERCEIVED WHOLES

The key idea of the proposed method of part embedding in Chapter 2 is the concept of *a perceptual whole* which replaces analytic abstractions (lines, parametric curves) and symbolic abstractions (predefined parts) [28]. The boundaries of perceptual wholes are determined according to a mapping to the lower level label algebra and wholes are extracted accordingly and represented temporarily with over-complete graphs.

The boundaries of the perceptual wholes are utilized as registration marks during the part searches. For the registration marks to be effective in the searches, both the whole shape and the searched part should agree in the manner registration marks are picked. Since no regularity is being imposed on the nature of parts, regularity should be imposed on the nature of boundaries. In perception science, these two main approaches to shape are exemplified by the seminal works of Biederman ([7]) in which the regularity is imposed on the parts and Hoffman and Richards ([24]) in which the regularity is imposed on part boundaries. Although a default marking mechanism is provided for determining the registration marks in Chapter 4 for a generic solution, the idea depends on respecting and supporting the user's/designer's choice for the registration marks to merely reflect the intention of the user/designer. In practice, this puts an additional burden on the designer, forcing his or her to imagine better towards the finished product.

In this chapter, a new approach to the part embedding is presented in order to eliminate the tedious procedure of marking. The key idea is letting all of the points coincident with the design plane - even the ones which are not on the design - serve as registration marks. To this end, the proposal is based on transforming a U_{12} shape to a weighted U_{22} shape where the weights indirectly capture the interaction among shape parts. The idea extends to U_{33}

as well when not only the design plane but the design volume is taken into consideration as a weighted shape. In this set-up, distinctions between curved or line shapes also become irrelevant.

The proposed method in this chapter for embedding parts has two main contributions. The first contribution is the representation of the drawings with the weight functions where the weights indirectly capture the interaction among shape parts during the embedding computations. In this way, we defer from relying solely on the boundaries in algebra $U_{(i-1)j}$ and implement the algebraic formalization using algebra $U_{(i+1)j}$. The key idea is that we let not just the boundaries of the design but the entire design plane to serve as a registration mark. The second contribution is the utilization of a genetic algorithm to compute a transformation that embeds a given part within a given whole.

3.1 Mapping U_{12} to Weighted U_{22}

Let Ω be an open, connected, bounded set representing the canvas on which a U_{12} shape, S, has been drawn. Let I_s be the indicator function whose value at a point is either one or zero depending on whether the point is on the shape S. Naturally, $\Omega \supset S = \{p : I_s(p) = 1\}$. We remark that the set Ω is another representation for the shape which may be thought as a dense U_{02} form.

Let us define a U_{22} shape which coincides with the set Ω. It is assumed here that the canvas is isolated and a weight is assigned to every point on it according to *the edge strength function* ([71]), which is obtained by solving the following reaction-diffusion equation. In this context, we will refer to the edge strength function as the weight function.

$$\left(\nabla^2 - \frac{1}{\rho^2}\right)\omega_s(\vec{p}) = 0 \tag{3.1}$$

where ∇^2 is the Laplace operator and ρ is a constant. This equation is subject to the following boundary conditions:

$$\frac{\partial \omega_s(\vec{p})}{\partial n} = 0, for\ \vec{p} \in \partial\Omega \tag{3.2}$$

and

$$\omega_s(\vec{p}) = 1, for\ \vec{p} \in S. \tag{3.3}$$

The weight function ω_s is one along the shape boundary. On the rest of the canvas it rapidly decays as a function of the distance from shape points. It is a smoothed version of the distance transformation. It is precisely this nature that captures a global percept of the shape as it implies strong interaction among shape points. This equation is implemented using the standard finite difference approximations (Appendix-B).

Figure 3.1 illustrates a sample U_{12} shape and its corresponding weighted U_{22} shape.

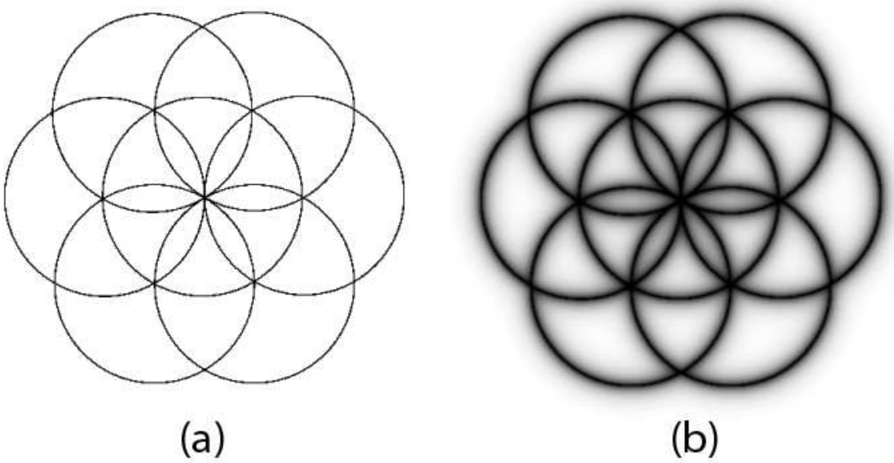


Figure 3.1: (a) A U_{12} shape, (b) Corresponding weighted U_{22} shape.

Both the original drawing S on the digital canvas Ω and the function ω on Ω can be thought as dense U_{02} shapes. Since the number of points is not critical in the proposed method, it is considered here as a continuum of infinitely many points.

The parameter ρ controls the amount of smoothing in the reaction diffusion equation. Its value is set considering the shape size and line characteristics of the shapes. The images that are used during the experiments in the context of this study have dimensions in the range of 400 by 400 and 600 by 600 pixels. In general, if the smoothing factor, ρ, is set to a too large value in these image dimensions, i.e. more than 16, the dominant features along the lines would fade away. Especially for the images which are composed of lines located very close to one another there would be too much interference of the weights in the regions that fall among these lines. All these factors affect the alignment precision of the lines during embedding. For these types of images ρ is set to 4 in order to avoid such interference. For the images which has wide empty spaces among the lines, ρ is initialized as 8.

3.2 Embedding In Weighted U$_{22}$

After mapping the shape from U_{12} to the weighted U_{22}, the embedding problem can be redefined as that of finding the transformation T that minimizes the function given below:

$$\int_{\Omega} \alpha(\vec{p})[\omega^W(\vec{p}) - (T \circ \omega^S)(\vec{p})]^2 d(\vec{p}) \tag{3.4}$$

Where ω^W and ω^S are the weight functions for the whole shape and the part, respectively; $T \circ \omega^S$ is the transformed weight function of the part. In this formulation α is a threshold function which in the cost calculation eliminates the contribution from far away points where the weight values are below a certain threshold. In all the experiments we provide in this chapter, the threshold value is selected as 0.5.

In this work, it is assumed that the transformation is global and can be parameterized by scale, position and orientation which are computed as minimizers of the cost function given in 3.4. In the discrete case, this energy can be expressed in terms of the transformation parameters as follows:

$$minE(h, t_x, t_y, \theta) = \sum_{i,j \in \Omega} \alpha_{i,j}[\omega^W_{i,j} - (T_{h,t_x,t_y,\theta} \circ \omega^S))_{i,j}]^2 \tag{3.5}$$

where h, t_x, t_y, θ are the transformation parameters for scale, position in x coordinates, position in y coordinates and orientation, respectively. In this formulation $(T_{h,t_x,t_y,\theta} \circ \omega^S)_{i,j}$ is the value of the transformed weight function of the sub-shape at (i, j). Assuming that each point on the canvas Ω which is coincident with ω^S is represented in the column vector form, the transformation T can be represented in the matrix form in homogeneous coordinates as follows:

$$T_{h,t_x,t_y,\theta} = \begin{bmatrix} hcos\theta & -hsin\theta & t_x \\ hsin\theta & hcos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

3.3 The Genetic Algorithm Implementation using MATLAB Toolbox

The minimization of the energy given in the equation 3.5 is performed using the Genetic Algorithm (GA) with some constraints defined for each optimization parameter. Each run of the GA operates on this set and range of parameters. It produces a possible embedding when ranges are determined in advance to limit the search space.

The GA is constructed to run on a population consisting of a finite number of individuals each of which is a candidate solution for the minimization problem. Each individual is represented with a genotype of four parameters in the chromosome structure. A genotype is mapped to a phenotype during the evaluation of the individuals. In this implementation a phenotype contains four decision parameters, one stands for scale, two for position and one for orientation.

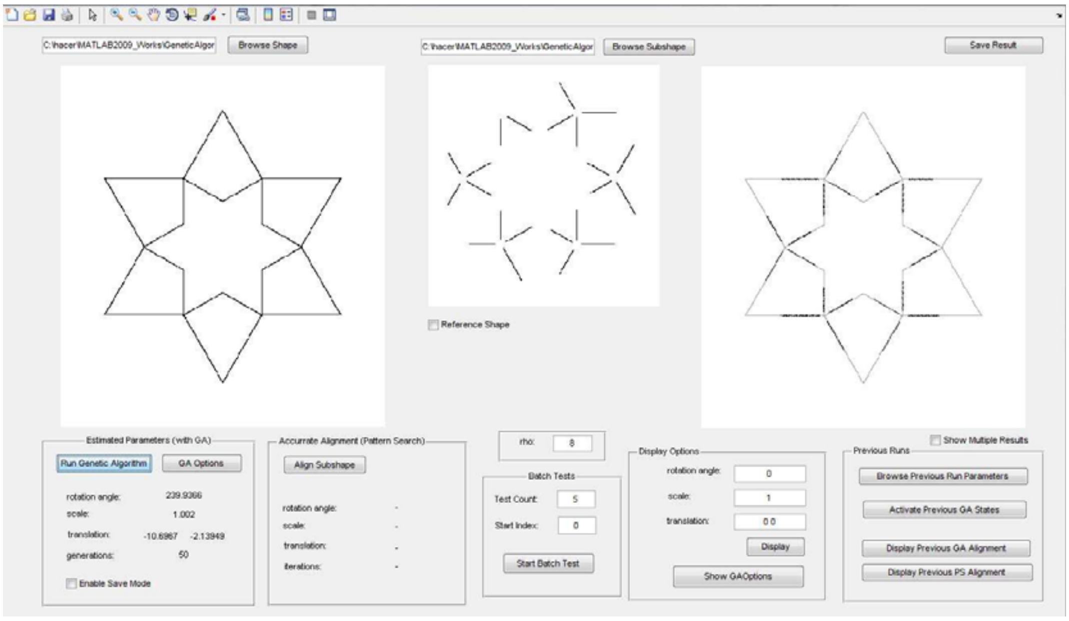
During the minimization, an objective function is used to evaluate the performance of the individuals (or phenotypes). The objective function, in the course of this work is inversely proportional to the cost function given in the equation (3.4). The fitness values are derived form this objective function values according to a rank based scaling.

At each evolution step, parents are selected for mating using a stochastic uniform sampling (SUS). SUS is a fitness proportionate (or a roulette wheel) selection where several solutions from the existing population are selected by repeated random sampling. In all the experiments provided in this chapter, parents are selected using SUS, thus using a roulette wheel algorithm and uniform sampling.

Following the mating, information is exchanged between potential individuals to generate offsprings by means of cross-overs. In the experiments, the cross-over rate is set as 0.6 to increase the diversity of the population at each evolution. In the part embedding problem, there may be multiple solutions, hence there are usually more than one local minima. This is the case especially when the shape is composed of repeated occurrences of some geometric primitives. This cross-over rate makes it possible to identify different local minima for different iterations. During the cross-overs, the parameters of the offsprings are set by a heuristic method. In this method, each parameter is computed by moving in the direction from worst parent's parameter to the best parent's parameter and set to a value which slightly passes the best parent's.

After the cross-over, the parameter values are modified with mutations with a probability of 0.4. Since this is a constrained GA, during the mutations, the parameters are guaranteed to be within the defined boundaries.

The part embedding application with the genetic algorithm is implemented using MATLAB v7.8 on Windows OS with a graphical user interface (GUI) support. A snapshot of the application GUI is as shown in Figure 3.2.



C:\hacer\MATLAB2009_Works\GeneticAlgor

C:\hacer\MATLAB2009_Works\GeneticAlgor

☐ Reference Shape

Estimated Parameters (with GA)
rotation angle: 239.9366
scale: 1.002
translation: -10.6967 -2.13949
generations: 50
☐ Enable Save Mode

Accurate Alignment (Pattern Search)
rotation angle: -
scale: -
translation: -
iterations: -

rho: 8

Batch Tests
Test Count: 5
Start Index: 0

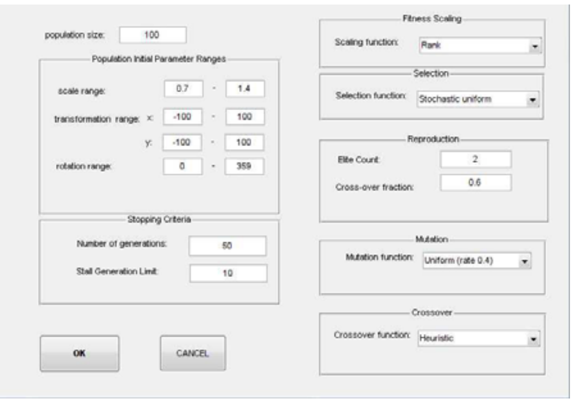
Display Options
rotation angle: 0
scale: 1
translation: 0 0

☐ Show Multiple Results

Figure 3.2: Part Embedding GUI

The initial shape which is introduced by the user in bitmap form is loaded into the left-most window of the GUI. The part shape which will be embedded within the initial shape using this application is shown in the middle window. The implementation supports single run as well as multiple runs all started at once. After the evolution of the GA is completed, the embedding of the part shape within the initial shape is shown in the right-most window of the GUI. The embedded part is depicted in bolder lines with respect to the initial shape as shown in the right-most window of the Figure 3.2.

In part embedding, the GA implementation considers the parameters specified via a GUI which we referred to as the GA Profiler. This GUI is embedded within the main application GUI and provides some options for many GA functions and parameters as shown in Figure 3.3.



population size: 100

scale range: 0.7 - 1.4

transformation range: x: -100 - 100

y: -100 - 100

rotation range: 0 - 359

Number of generations: 50

Stall Generation Limit: 10

Scaling function: Rank

Selection function: Stochastic uniform

Elite Count: 2

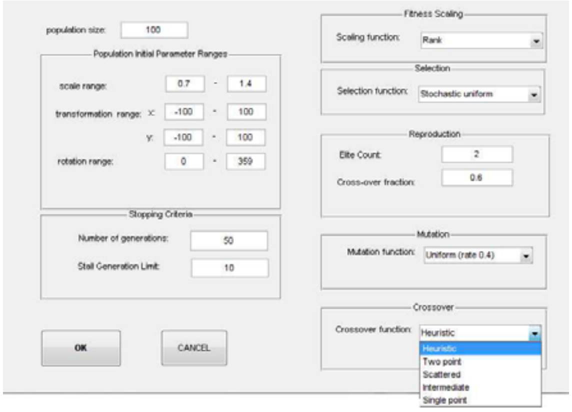
Cross-over fraction: 0.6

Mutation function: Uniform (rate 0.4)

Crossover function: Heuristic

Figure 3.3: GA Profiler GUI

Not only the parameters with scalar values, but also the functions which directly influence the evolution like the selection, mutation and cross-over can be set per experiment basis using this GUI. A sample option set for the cross-over function provided with the GA Profiler is shown in Figure 3.4. The other GA functions can be set in similar fashion.


population size: 100

Population Initial Parameter Ranges
scale range: 0.7 - 1.4
transformation range: x: -100 - 100
y: -100 - 100
rotation range: 0 - 359

Stopping Criteria
Number of generations: 50
Stall Generation Limit: 10

Fitness Scaling
Scaling function: Rank

Selection
Selection function: Stochastic uniform

Reproduction
Elite Count: 2
Cross-over fraction: 0.6

Mutation
Mutation function: Uniform (rate 0.4)

Crossover
Crossover function: Heuristic

Figure 3.4: The options provided with the GA Profiler GUI for the cross-over function in addition to the Heuristic method are Two-Point, Scattered, Intermediate and Single-Point cross-over functions.

In the experiments, the number of individuals in a population for the runs is typically set to 100. Individuals of the population are represented with four parameters, two parameters for transformation (in x and y axes), one for orientation and one for scale. In this constrained GA implementation, the constraints for the parameters can be defined by the user via GA Profiler GUI. The constraints pre-defined for the scale parameter is crucial. Unless a lower bound to the scale parameter is given, the best alignment for any run would be obtained when the shape is scaled down until it fits to one or a couple of pixels. The embedding cost becomes small enough for the shape to be aligned with any part of the initial shape. In our experiments the lower bound for the scale parameter is set as 0.7.

3.4 Advantages of Using Weights In Embedding

In [28], points serve as registration marks in mapping and they are all identical except for the attributes assigned for matching. In this approach, however, points are not identical any more, since each point has a corresponding weight value which makes it unique. The relations of a point with its surrounding points encode the properties of that neighborhood and mark the unique location of that point within the shape. The weight function, in this respect, provides underlying biases within a shape that come in handy when matching two shapes through alignment.

Assume that in the equation (3.5), instead of the weight functions $\omega^W_{i,j}$ and $\omega^S_{i,j}$, we use the drawings represented with the indicator functions $I^W_{i,j}$ and $I^S_{i,j}$, respectively. In that case, aligning two line shapes - two sets of pixels in the discrete domain - by minimizing the equation 3.5 with GA is prone to be trapped in the false local minima. This can be overcome by utilizing the supplementary information provided by the weight function which implicitly codes the local characteristic of a shape.

The role of the weight function for successful part alignments becomes obvious when we make some experiments using weighted shapes instead of the given initial shapes. For this purpose, a number of experiments are performed to make sure that embedding with weighted shapes give better results than embedding with the given shapes. For a selected initial shape and a part shape, we depict here the embedding results of each experiment sets. Both of these sets of experiments are performed for 10 successive iterations of the GA using exactly the same configuration settings. In the experiment set where we used the indication function directly for embedding, the resultant part alignments show that only 20% of the alignments are close to one of the expected solutions (Figure 3.5).

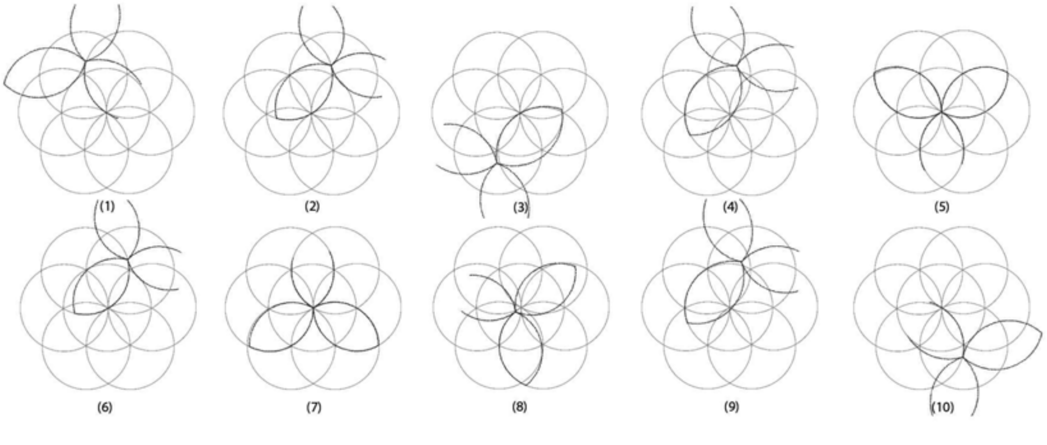

(1) (2) (3) (4) (5)
(6) (7) (8) (9) (10)

Figure 3.5: Part alignments of 10 successive iterations of the GA without using weights. Except for the iterations (5) and (7), the alignments are far from expected results. In these experiments the following GA parameters are used: population size = 150, number of generations = 70, scale range = 0.9-1.4, transformation range = [-70,70], rotation range = [0,359], cross-over fraction= 0.6, elite count = 2.

The same algorithm is run for 10 successive iterations, with the same configuration settings for the GA, using the weighted shapes and the resultant embedded part alignments are as shown in Figure 3.6. In these experiments, the weight function is computed with $\rho = 8$ and the algorithm successfully aligns the part within the given initial shape with a rate of 90 percent. When the number of successive experiments is increased to 50, the success of the algorithm is approximately around 85%.

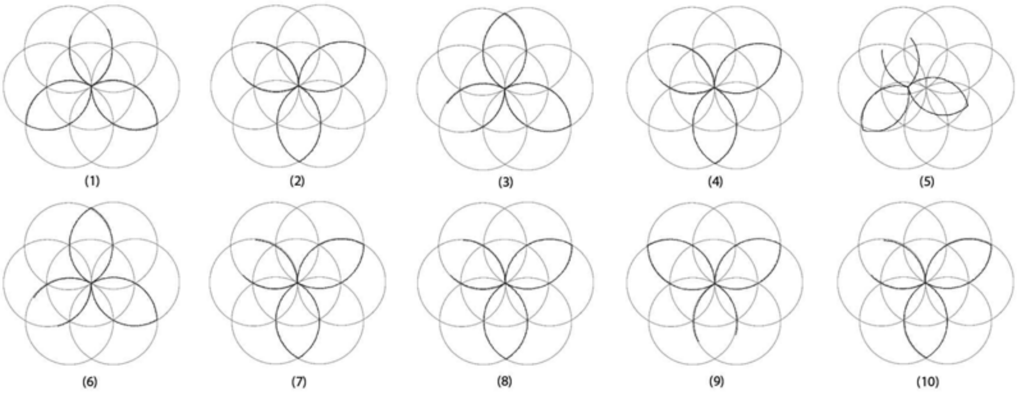


Figure 3.6: Part alignments of 10 successive iterations of the algorithm with weighted images. Except from experiment (5), all the alignments are very close to the one of the expected solutions. In these experiments the following parameters are used in GA: population size = 150, number of generations = 70, scale range = 0.9-1.4, transformation range = [-70,70], rotation range = [0,359], cross-over fraction= 0.6, elite count = 2.

These experiments show that, the weight functions provide significant information to embed one shape within another by using the constraints within the shapes. This is advantageous since there is no need to define external constraints for matching. At the same time, the weight functions avoid pre-defined decompositions and stay true to the continuous nature of shapes.

3.5 No Presumptions About Shapes

In the shape grammar literature, curved and linear shapes technically call for different measures in implementation. The work of Chau et. al. [14] is based on maximal representation which considers maximal circular arcs in planes and space under Euclidean transformation. McCormack and Cagan's implementation [45] on the other hand supports free-form curves by converting freeform curves to the shapes composed of straight line segments. The recent work of Jowers and Earl [27] for developing a shape grammar interpreter is based on an intrinsic matching algorithm for shapes composed of parametric curve segments. Each of these works relies on distinct notions of curved lines. The advantage of the proposed approach here is that shapes are treated as they are so that any presumed analytic abstractions related with shapes, like curved shapes or linear shapes, are avoided.

The presented method is indifferent to such abstract representations. Hence, any distinctions among them become irrelevant. Shapes that combine different type lines are easily handled as all are treated as perceived wholes (Figure 3.7 and 3.8). The matching applies to all shapes.

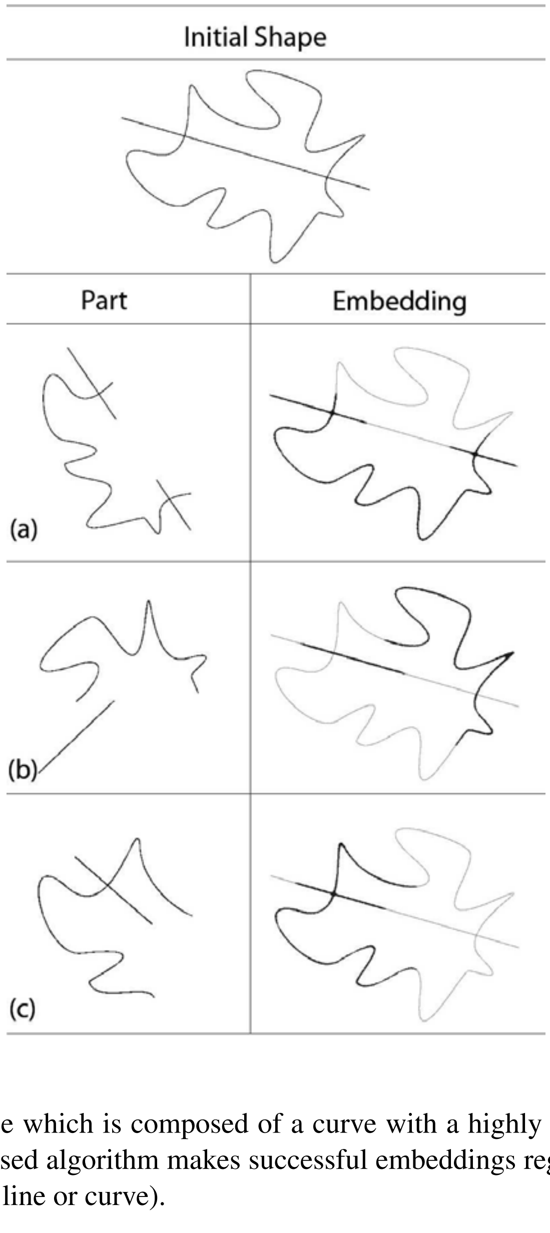


Figure 3.7: For a shape which is composed of a curve with a highly free-form nature and a straight line, the proposed algorithm makes successful embeddings regardless of the types of its basic elements (i.e. line or curve).

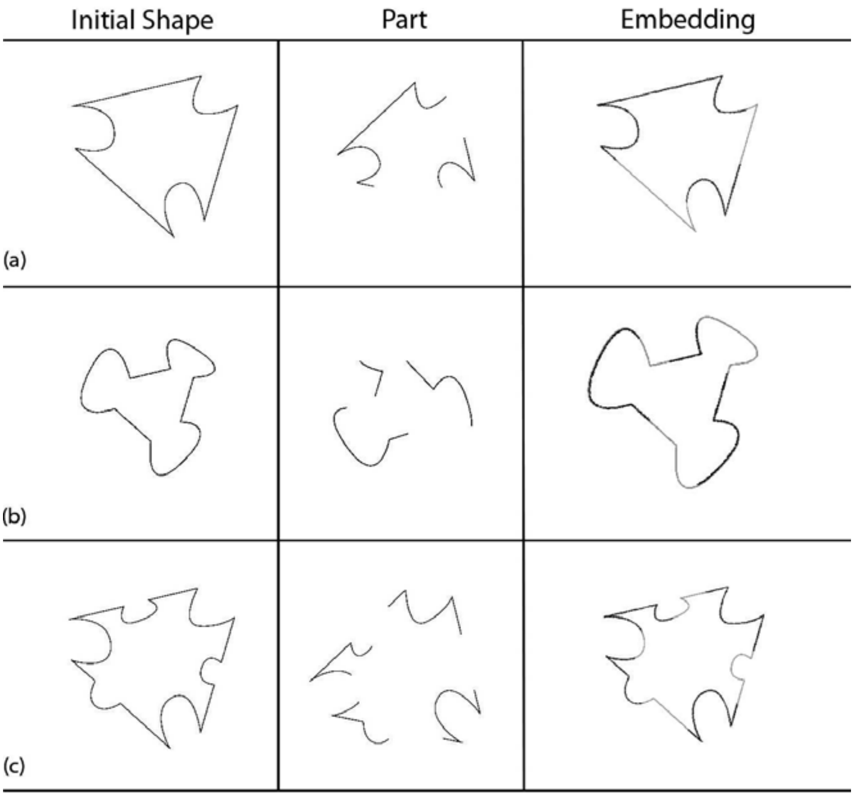


Figure 3.8: The sample shapes are composed of straight lines and curves connected end to end. The estimated four transformation parameters, one for orientation, one scale, two for transformations (in x and y coordinates) with the GA for (a) is (330, 1.25, -1.7, 34.2); for (b) is (300, 0.98, 13.4, 11.7) and for (c) is (329,0.99,-2.16,19.4).

The success of the proposed method for any shape is visible as we make some experiments with shapes in different characteristics. For an initial shape which is composed of curved lines, the proposed algorithm detects all possible embeddings for the given part shapes under the Euclidean transformations as shown in Figure 3.9.

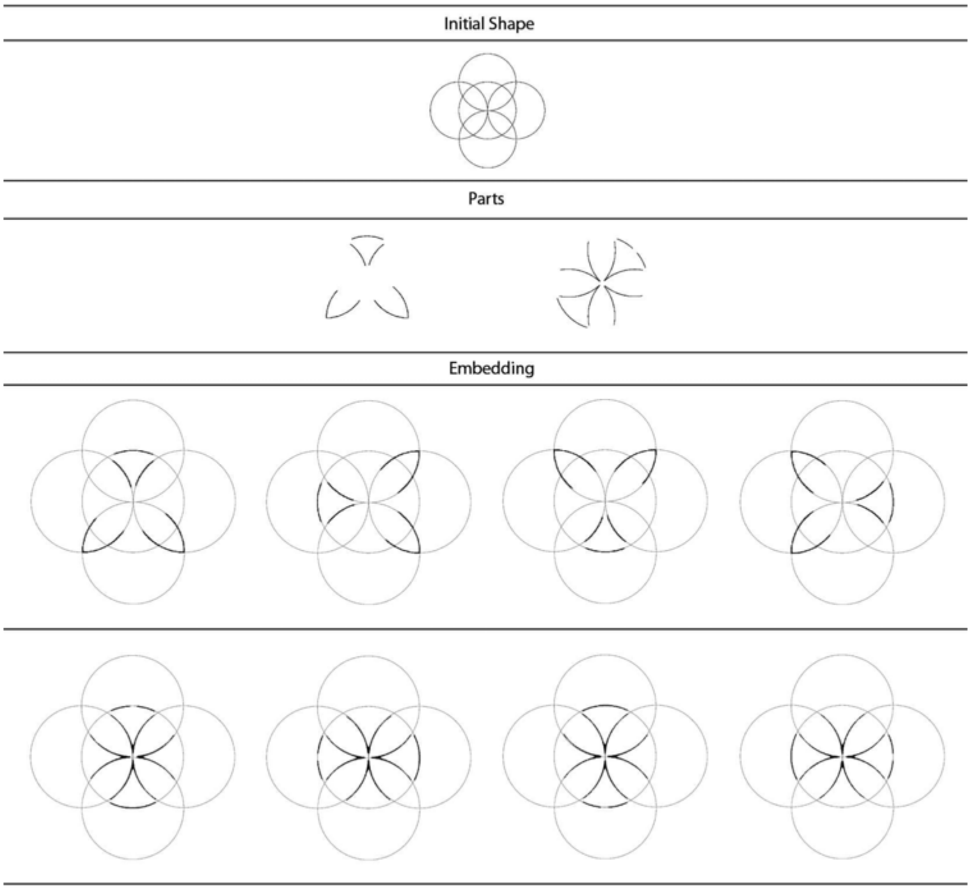


Figure 3.9: All different part embeddings that are obtained with the proposed algorithm is depicted for two sample curved part shapes.

Similarly, for an initial shape that has many symmetry axes and hence contains recurrent structures inside, all possible parts under the Euclidean transformations are easily detected (Figure 3.10). This time, the shapes used in the experiments are composed of straight lines.

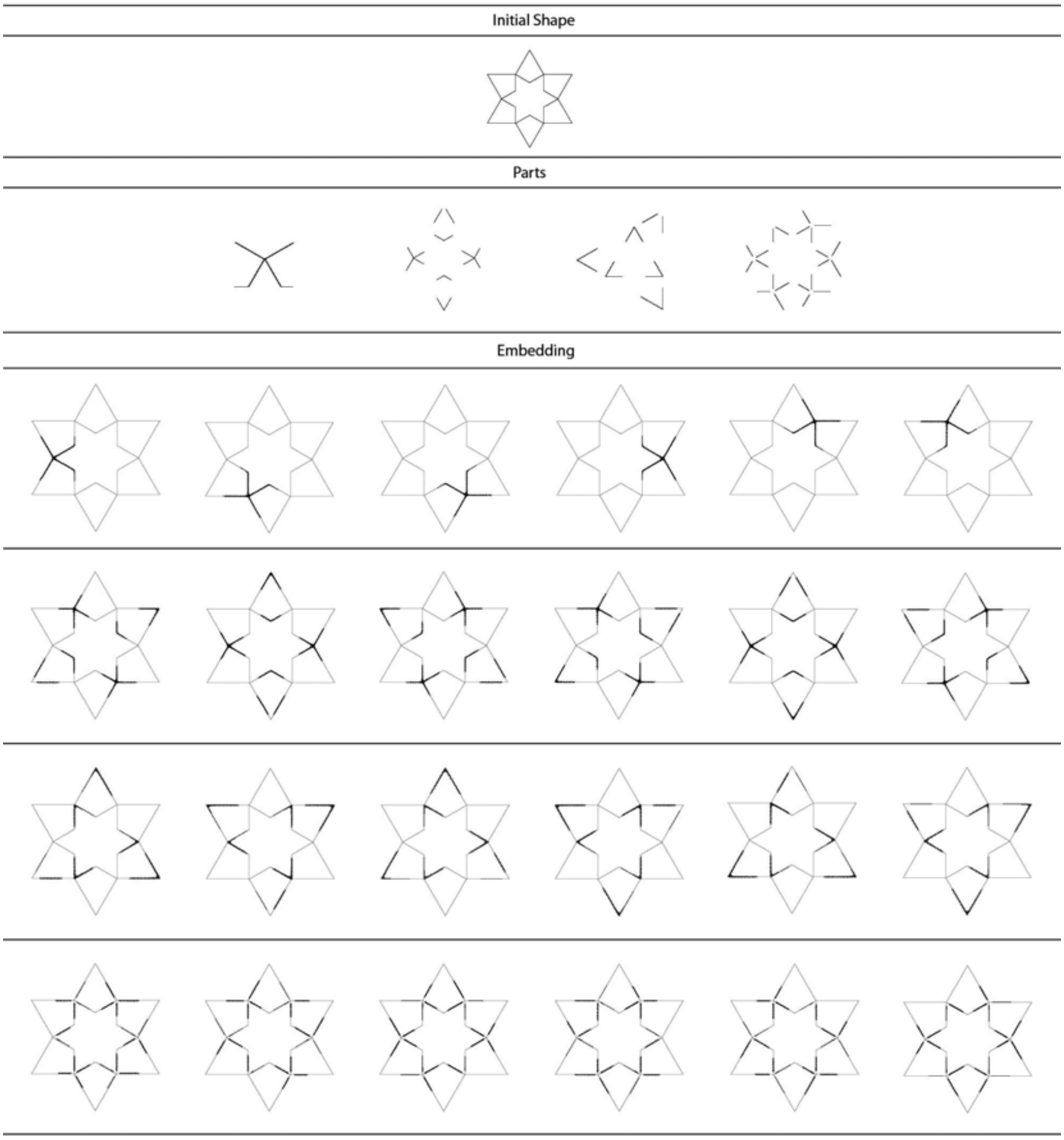


Figure 3.10: All different part embeddings that are obtained with our method is depicted for four sample part shapes with straight lines.

These sample runs show that the performance of the proposed algorithm for embedding parts is indifferent whether the shapes are of straight or curved lines.

Since the shapes are not interpreted in a predefined manner, there is no restriction in the definition of the part shapes in this method. This implies that, the left hand side shapes of possible visual rules - shown in the tables as parts searched in the initial shape - can be defined in any desired way (Figure 3.11). All these parts can easily be detected within the initial shape.

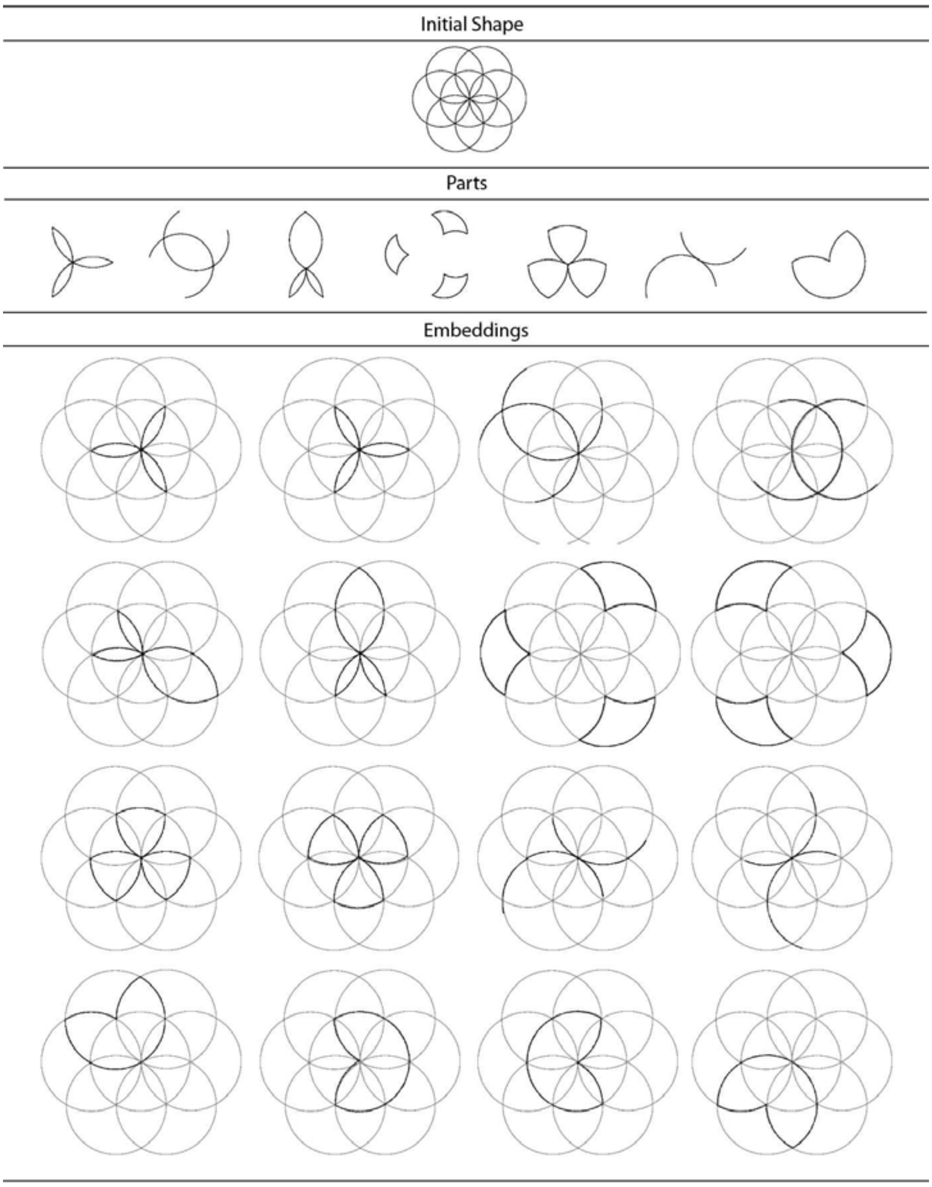


Figure 3.11: Sample embeddings of a set of different parts within the same initial shape.

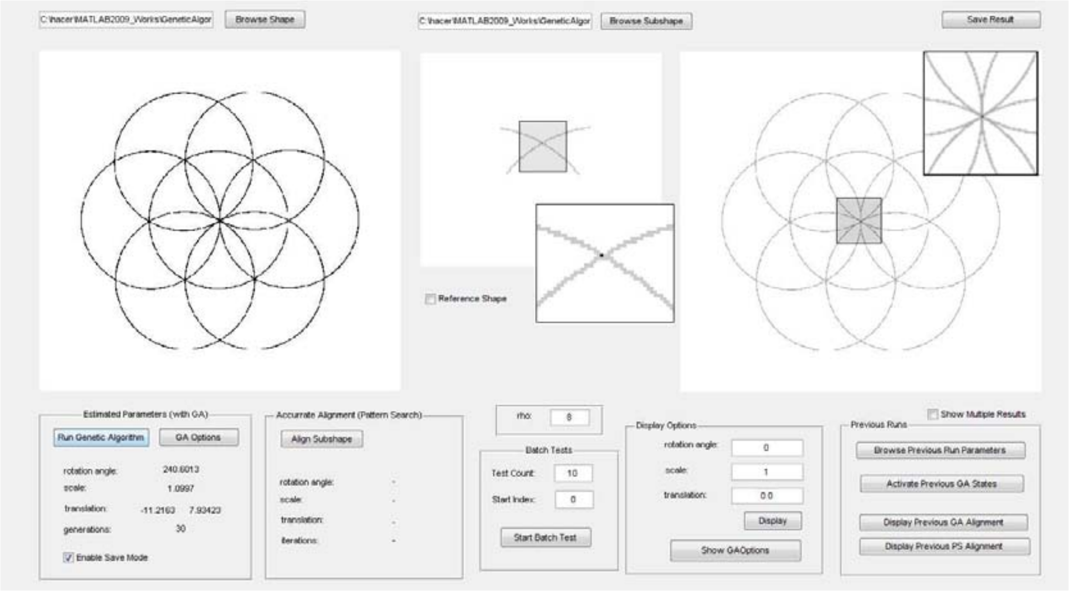
3.6 Experiments And Discussion

For the experiments provided in this section, geometric line drawings are employed in the style of Seljuk patterns. These patterns contain repetition of abstract line configurations derived by transforming the various grids underlying the design. There are various reasons why we have chosen these patterns. One reason is that this particular pattern building process contains emergent shapes as the shape context and perception constraints change along the way [5]. Another reason is that understanding the visual computations behind the style relies on understanding the perceived wholes that guide the construction of the next step. Moreover, understanding such patterns as a visual rule based construction could be generalized to many other traditional productions as suggested by Knight [31]. Ozkar and Lefford have already partially addressed the process based assessment of Seljuk and Celtic pattern styles [48].

The method allows for part matching in a variety of ways as discussed in the upcoming subsections.

3.6.1 Embedding for Visual Rule Based Construction

For the experiments in this part, we use a set of visual rules. The work presented in Chapter 2 which is based on points as registration marks is advantageous when shapes are decomposed from their intersecting points. In such cases, the proposed approach in this chapter also allows for the detection of points for the purpose of picking out such parts. The reference shape concept [28] is still very useful in this approach too, to embed ambiguous parts like points in an unambiguous fashion according to the given reference shape/context. Hence, any point we want to see within a shape may be specified with the help of a proper reference shape and in a context-based search. For example, in Figure 3.12, a part of the two curve crossings is used as a reference shape to embed a point that is located at the intersection of these two curves.



C:\hacer\MATLAB2009_Works\GeneticAlgor

C:\hacer\MATLAB2009_Works\GeneticAlgor

☐ Reference Shape

Estimated Parameters (with GA)

rotation angle: 240.6013
scale: 1.0997
translation: -11.2163 7.93423
generations: 30

☑ Enable Save Mode

Accurate Alignment (Pattern Search)

rotation angle: -
scale: -
translation: -
iterations: -

rho: 8

Batch Tests

Test Count: 10
Start Index: 0

Display Options

rotation angle: 0
scale: 1
translation: 0 0

☐ Show Multiple Results

Previous Runs

Figure 3.12: Embedding a point using a reference shape

In order to embed a point in this way, the embedding parameters estimated with the GA for the reference shape is used. Two samples are displayed in detail in Figure 3.13. In (a), the point in the central region of the shape is embedded according to the embedded reference shape displayed on the left. Similarly, in (b) another point of intersection is embedded owing to the embedded reference shape. As it is seen from these samples, the exact position of the embedded point is directly related with the position of the embedded reference shape. The reference shape provides an interface to the user to define the context easily as in [28].

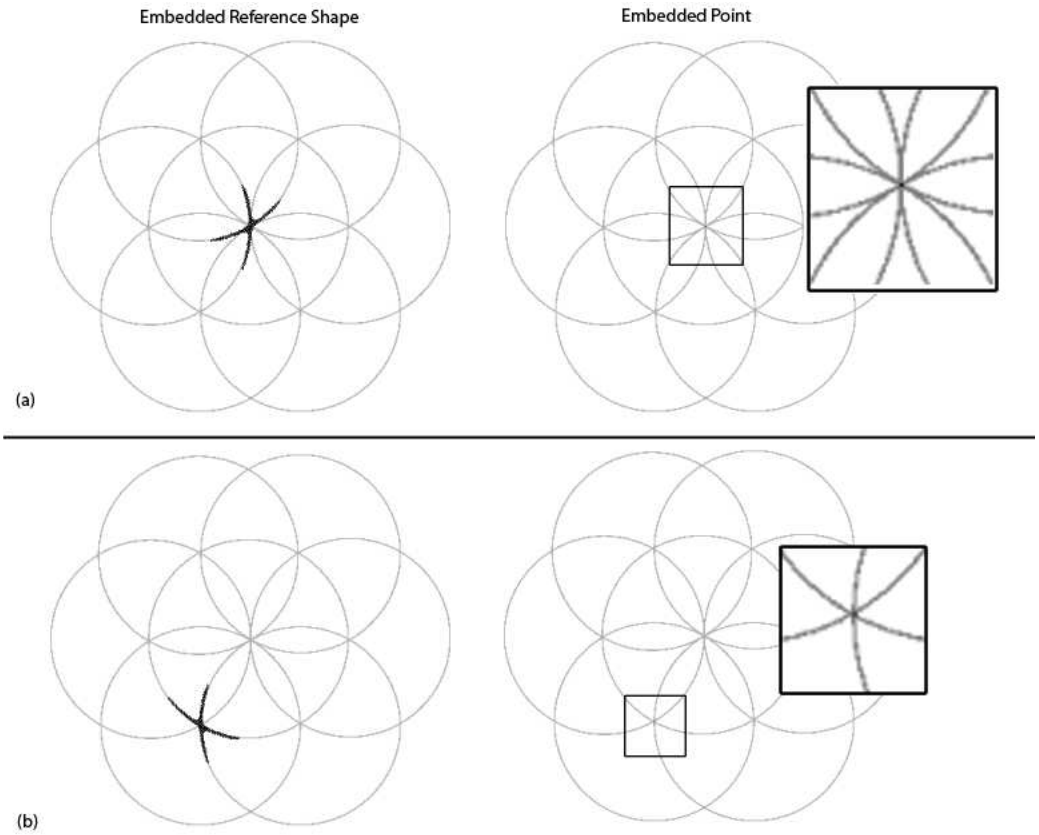


Figure 3.13: Two sample embeddings.

Assume that in a visual rule, we define the left hand side shape as a point with a reference shape as shown in the GUI in Figure 3.12 and embed some of the points in the initial shape (Figure 3.1-(a)).

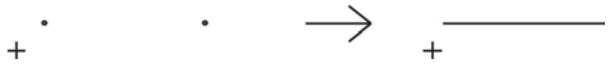


Figure 3.14: The visual rule which when applied combines two points with a line.

Using the embedded points, when we apply the visual rule shown in Figure 3.14, we can obtain new shapes which are composed of basic elements of both lines and curves. One such shape is depicted in (Figure 3.15-(a)). The corresponding weighted U_{22} shape which is used in the embedding computations when the shape in Figure 3.15-(a) is given as the initial shape is depicted in Figure 3.15-(b).

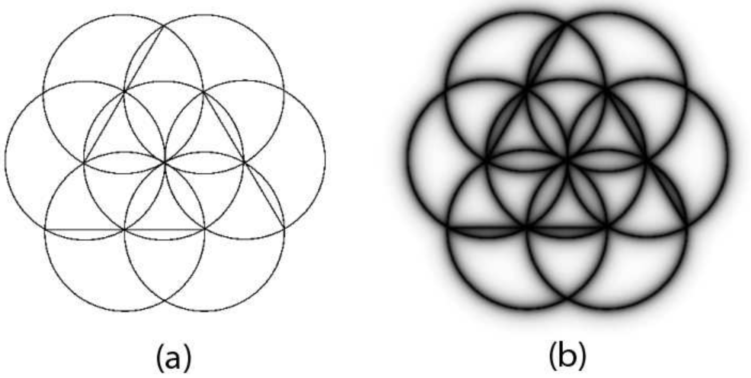


Figure 3.15: (a)The shape obtained after a number of rule applications to the shape in Figure 3.1, (b) Corresponding weighted U_{22} shape.

As it is mentioned in an earlier section, in this approach any distinctions between line and curve shapes disappear. Using the shape in Figure 3.15-(b), regardless of how the shape in Figure 3.15-(a) is created, we can now embed new parts which can be composed of basic elements of any type. Two such embeddings are depicted in Figure 3.16.

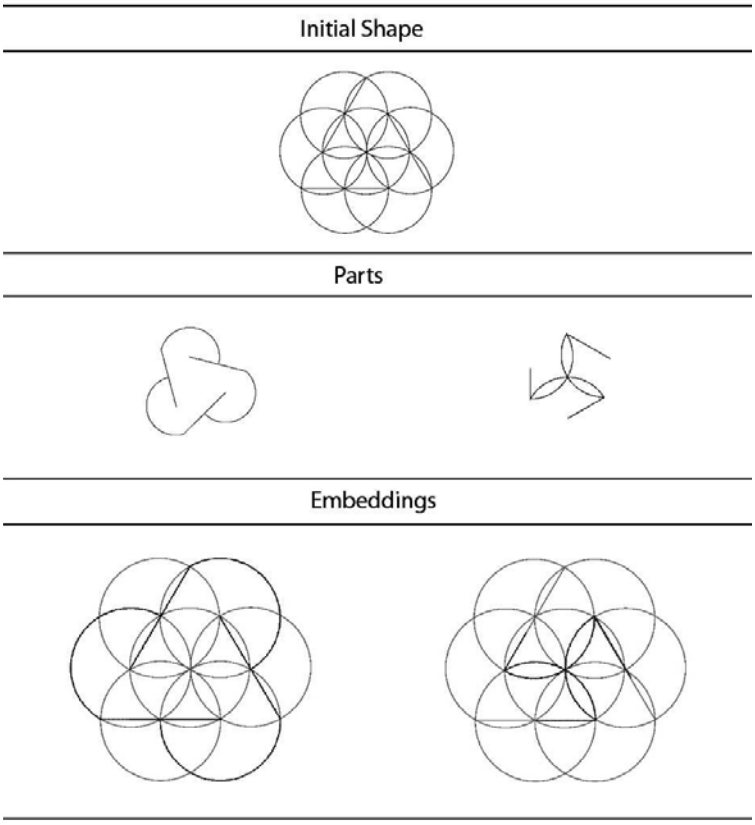
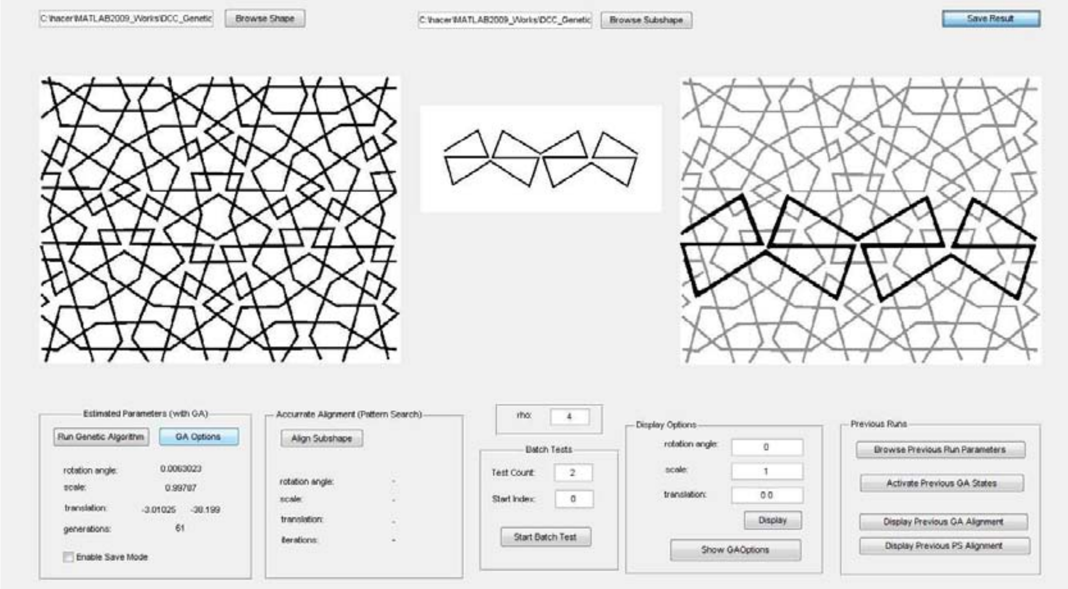


Figure 3.16: Sample part embeddings.

The construction of new shapes using some visual rules and using these newly generated shapes in embedding can go on this way forever. The proposed method requires no intervention or intermediate representation of the generated shapes except for the computation of the weights. Hence, the method is useful for visual rule based constructions for shape grammar interpreters.

3.6.2 Embedding within Planes

It is important to note that any distinctions between the abstract shape categories of line and plane also go away. The technique works well in picking out parts in crowded line drawings (Figure 3.17) or within planes (Figure 3.18 and 3.19). Notice that the embedding algorithm may reveal parts that are not drawn explicitly on the canvas as the one shown in Figure 3.18.


Browse Shape
Browse Subshape
Save Result
Estimated Parameters (with GA)
Run Genetic Algorithm
GA Options
Accurrate Alignment (Pattern Search)
Align Subshape
Batch Tests
Start Batch Test
Display Options
Display
Show GAOptions
Previous Runs
Browse Previous Run Parameters
Activate Previous GA States
Display Previous GA Alignment
Display Previous PS Alignment

C:\hacer\MATLAB2009_Works\DCC_Genetic

C:\hacer\MATLAB2009_Works\DCC_Genetic

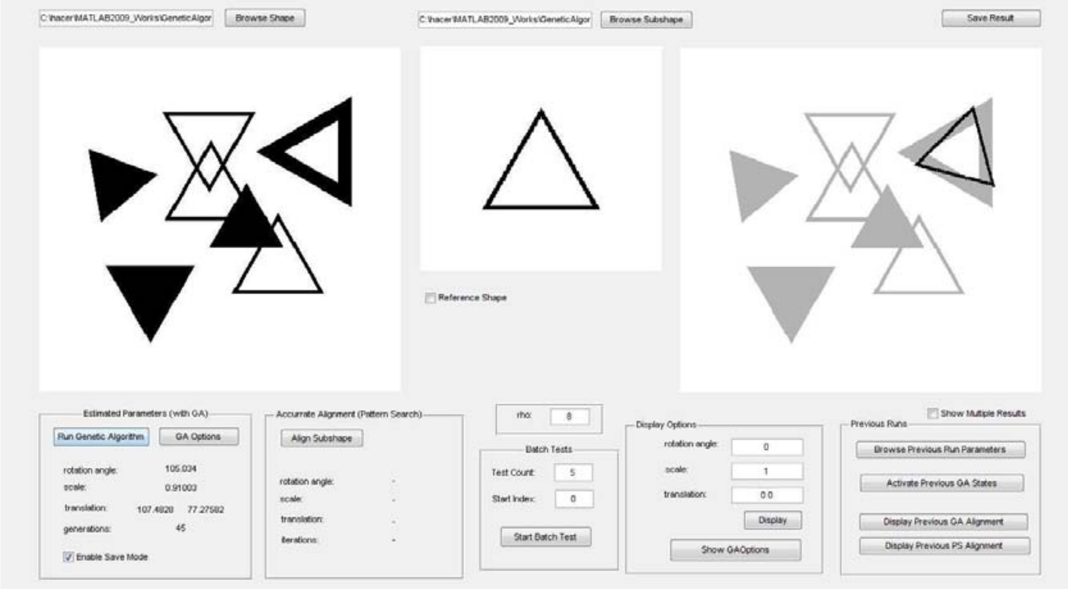
rotation angle: 0.0063023
scale: 0.99787
translation: -3.01025 -38.199
generations: 61
☐ Enable Save Mode

rotation angle: -
scale: -
translation: -
iterations: -

rho: 4
Test Count: 2
Start Index: 0

rotation angle: 0
scale: 1
translation: 0 0

Figure 3.17: Embedding a shape that is composed of parts of triangles into the line crowded Seljuk pattern.


Browse Shape
Browse Subshape
Save Result
Estimated Parameters (with GA)
Run Genetic Algorithm
GA Options
Accurrate Alignment (Pattern Search)
Align Subshape
Batch Tests
Start Batch Test
Display Options
Display
Show GAOptions
Previous Runs
Browse Previous Run Parameters
Activate Previous GA States
Display Previous GA Alignment
Display Previous PS Alignment

C:\hacer\MATLAB2009_Works\GeneticAlgor

C:\hacer\MATLAB2009_Works\GeneticAlgor

☐ Reference Shape

☐ Show Multiple Results

rotation angle: 105.034
scale: 0.91003
translation: 107.4820 77.27582
generations: 45
☑ Enable Save Mode

rotation angle: -
scale: -
translation: -
iterations: -

rho: 8
Test Count: 5
Start Index: 0

rotation angle: 0
scale: 1
translation: 0 0

Figure 3.18: A sample part embedding where the initial shape is composed of planar elements in addition to lines.

When the algorithm is run on the same initial shape shown in Figure 3.18, each run results in a different alignment. Some of these align the triangles in a similar way with the one shown in Figure 3.18, such as those in the top left and top right images of Figure 3.19. In all of these embeddings the part shape is aligned with the part of the initial shape in which some parts of the alignments are overlapping with the plane regions instead of the boundaries. For example in Figure 3.18, no line from the triangle corresponds to a boundary of the planar triangular shape on the canvas that it is picked out in. Such an embedding may be handy for a designer in the design process. It is because while working with the design sketches designers may prefer picking out emerging line shapes out of shaded areas or vice versa.

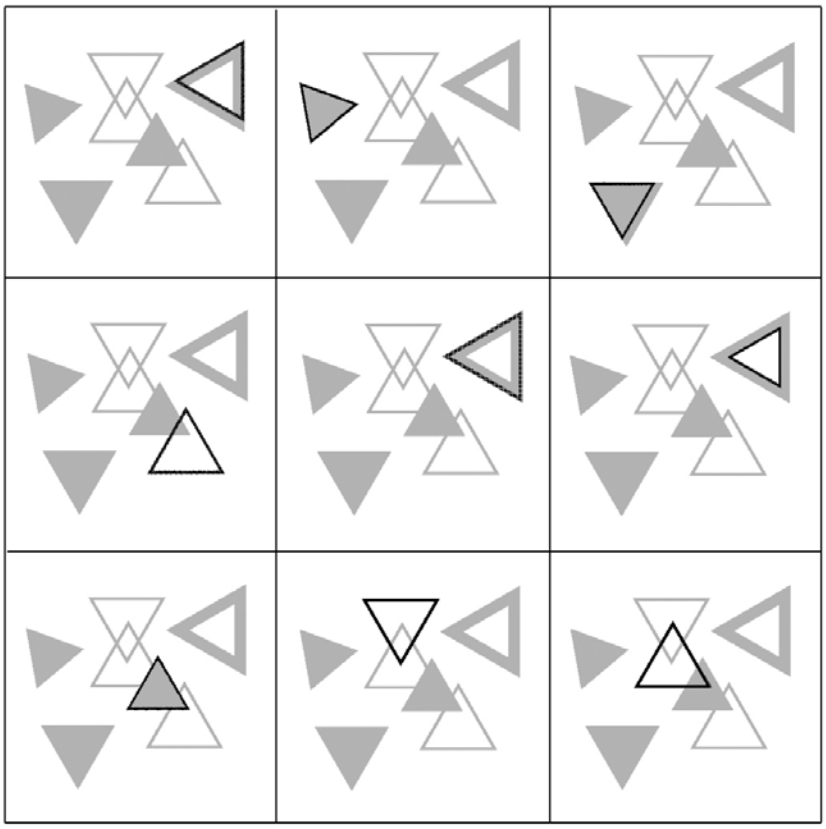


Figure 3.19: Sample part embeddings.

Consider the Seljuk pattern shown in Figure 3.20. Although at a first glance we see that the pattern is composed of lines, interpretation of this pattern with lines is difficult. This is because a line in this pattern has a neighboring line following it in parallel and the two together are actually the boundaries of a planar stripe.

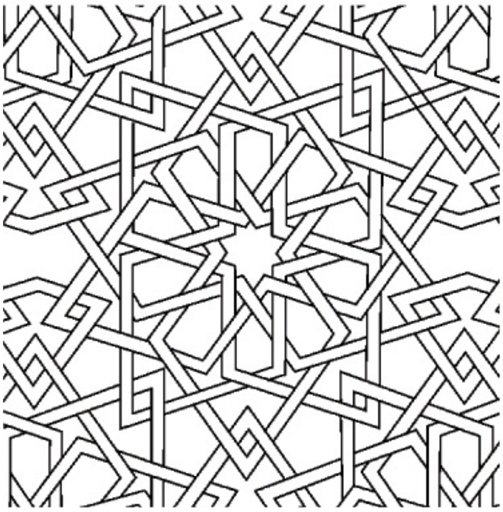


Figure 3.20: A Seljuk pattern from Huand Hatun Tomb (1237,1238), Kayseri, Turkey (drawn by Assoc. Prof. Dr. Mine Özkâr).

The artisan, on the original work in Kayseri, formed this craft by carving the stone throughout the surface with an apparatus of a specific width at its carving edge. The work appears as a composition of planar stripes in three-dimensions by going below/above of the other stripes all over the surface extending to infinity. This is more evident in Figure 3.21.

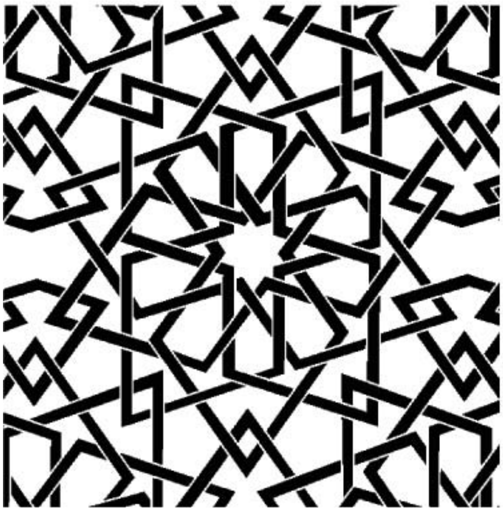


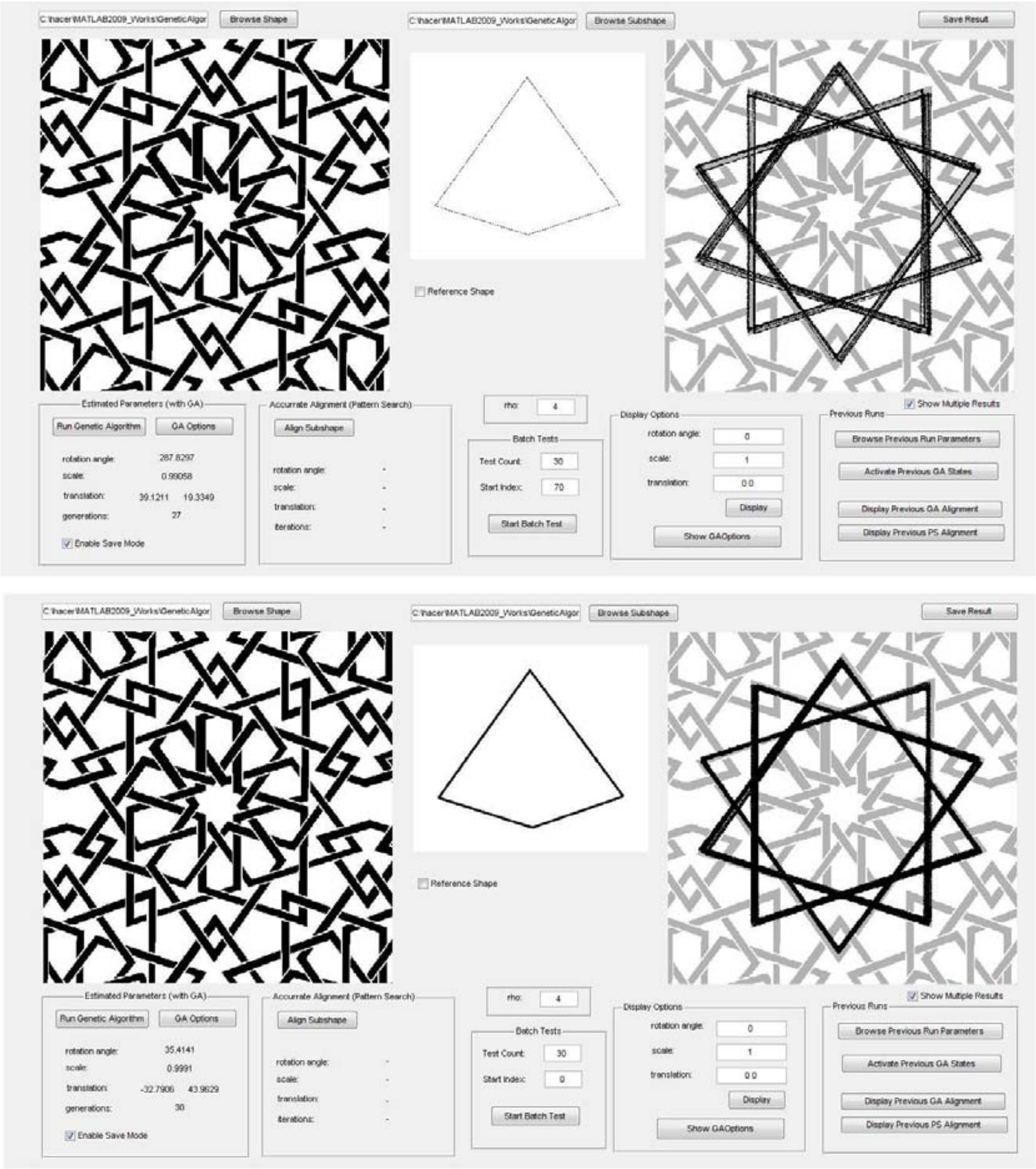
Figure 3.21: The Seljuk pattern in Figure 3.20 is redrawn with thick lines which are generated considering the weaving of the stripes.

This planar form represents the original work better. In this representation, the way stripes are woven makes interpretation of this pattern complicated. The weaving of the stripes introduces a depth to this pattern and projection of it on a planar surface results in broken lines [1], although they are parts of continuous stripes. This is why embedding within such shapes becomes challenging. This process requires a high-level interpretation of the shape, which would relate the broken lines as parts of a continuous stripe. It would probably be a custom solution for each such pattern, since most of the Seljuk patterns are rich in ambiguity. The approach proposed in this work, however, has no difficulty with working such patterns.

In an example, a quadrilateral shape is searched as a part of this shape. Whether the quadrilateral is drawn with thin lines or thick lines, the embedding succeeds (Figure 3.22).

---

[1] lines in the context of this pattern refer to parts of planar stripes which can be interpreted as thick lines.


Browse Shape
Browse Subshape
Save Result
Estimated Parameters (with GA)
Run Genetic Algorithm
GA Options
Accurate Alignment (Pattern Search)
Align Subshape
Batch Tests
Start Batch Test
Display Options
Display
Show GAOptions
Previous Runs
Browse Previous Run Parameters
Activate Previous GA States
Display Previous GA Alignment
Display Previous PS Alignment

C:\hacer\MATLAB2009_Works\GeneticAlgor
C:\hacer\MATLAB2009_Works\GeneticAlgor
☐ Reference Shape
rotation angle: 287.8297
scale: 0.99058
translation: 39.1211 19.3349
generations: 27
☑ Enable Save Mode
rotation angle: -
scale: -
translation: -
iterations: -
rho: 4
Test Count: 30
Start Index: 70
rotation angle: 0
scale: 1
translation: 0 0
☑ Show Multiple Results

C:\hacer\MATLAB2009_Works\GeneticAlgor
C:\hacer\MATLAB2009_Works\GeneticAlgor
☐ Reference Shape
rotation angle: 35.4141
scale: 0.9991
translation: -32.7906 43.9629
generations: 30
☑ Enable Save Mode
rotation angle: -
scale: -
translation: -
iterations: -
rho: 4
Test Count: 30
Start Index: 0
rotation angle: 0
scale: 1
translation: 0 0
☑ Show Multiple Results

Figure 3.22: (top) Embedding results of 100 distinct GA runs are aggregated, where the searched part is a one pixel-wide quadrilateral shape, (bottom) Embedding results of 30 distinct GA runs are aggregated, where the searched part is a six pixel-wide quadrilateral shape. The initial shape in both experiments is the planar representation of the Seljuk pattern depicted in Figure 3.21.

In the samples given above, notice the embedded parts of the initial shape. Although the stripes are broken in two-dimensional planar domain due to the crossing of another stripe along the way, these cuts become invisible and the part is embedded to reveal that part as a whole within the initial shape. This is very similar to how we perceive a continuous stripe instead of broken ones when we want to embed the quadrilateral shape there with our eyes.

The bottom figure in Figure 3.22 shows the aggregation of the 30 different runs. The aggregation steps for some selected runs are depicted in Figure 3.23. In each run, although the same GA configuration is used, different parts of the initial shape are embedded.

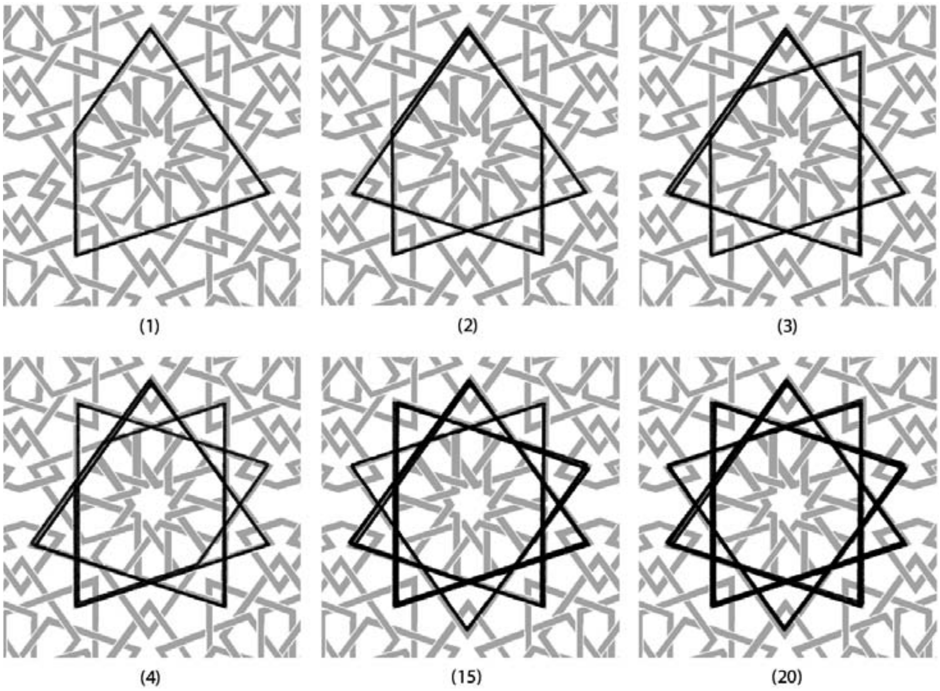
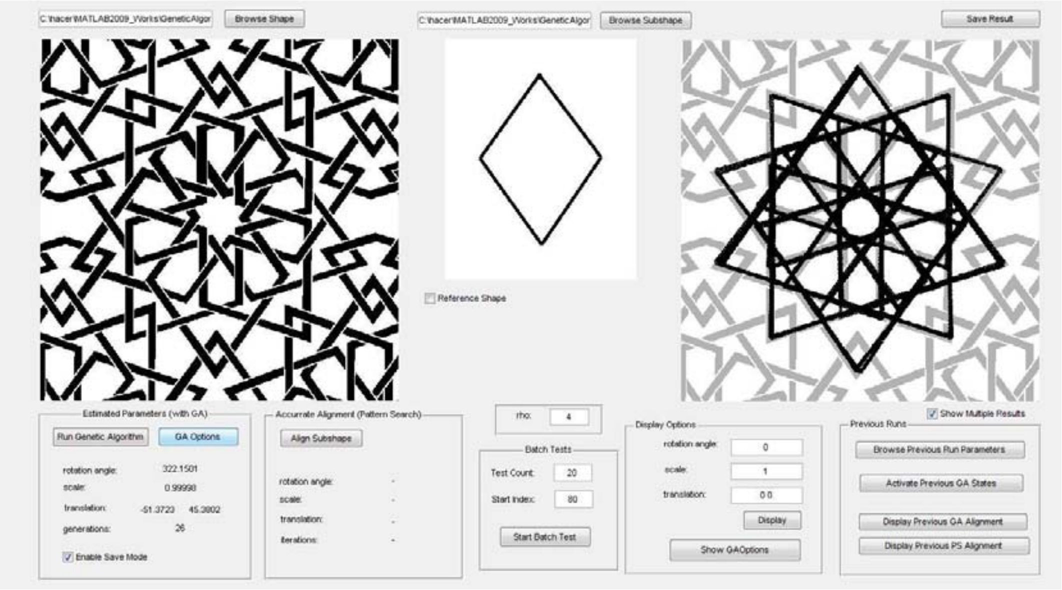


Figure 3.23: Each figure is formed by aggregating the embeddings of multiple runs. The number that is shown below each sub-figure stands for the index of the number of aggregated embeddings.

Similarly, when we aggregate multiple embedding results for the parts of the same Seljuk pattern with a rhombus, we obtain the shape shown below. This aggregation shows that the rhombus shape spans the central part of the Seljuk pattern completely. It is rather difficult to notice this at first when the initial shape is examined with bare eyes. In this respect, the algorithm may help analyzing the recursive patterns as this one.


Browse Shape
Browse Subshape
Save Result
Estimated Parameters (with GA)
Run Genetic Algorithm
GA Options
Accurrate Alignment (Pattern Search)
Align Subshape
Batch Tests
Start Batch Test
Display Options
Display
Show GAOptions
Previous Runs
Browse Previous Run Parameters
Activate Previous GA States
Display Previous GA Alignment
Display Previous PS Alignment

C:\hacer\MATLAB2009_Works\GeneticAlgor

C:\hacer\MATLAB2009_Works\GeneticAlgor

☐ Reference Shape

rotation angle: 322.1501
scale: 0.99998
translation: -51.3723 45.3802
generations: 26

☑ Enable Save Mode

rotation angle: -
scale: -
translation: -
iterations: -

rho: 4

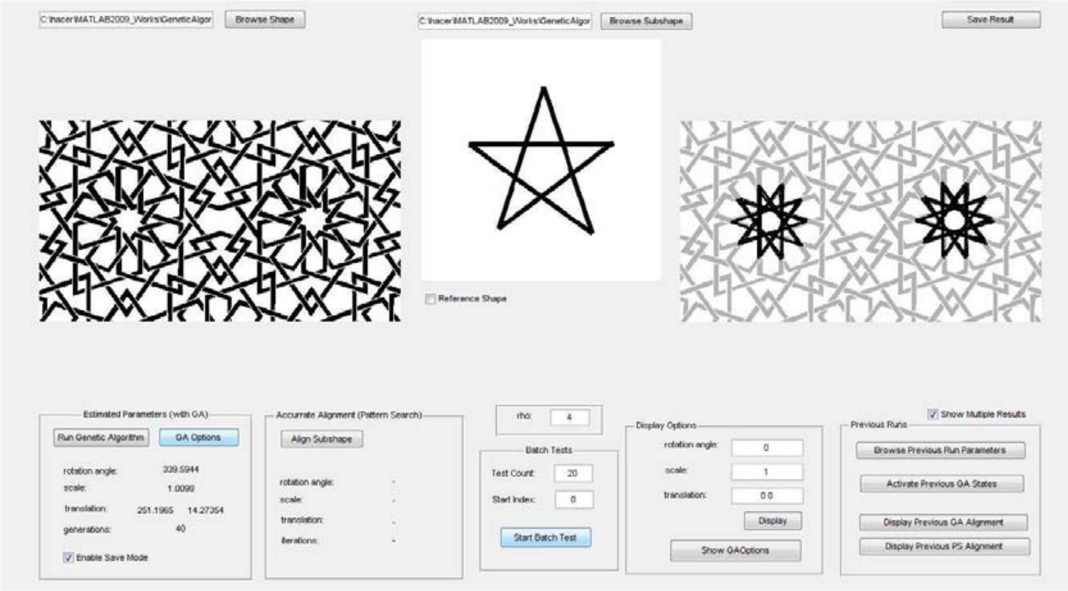
Test Count: 20
Start Index: 80

rotation angle: 0
scale: 1
translation: 0 0

☑ Show Multiple Results

Figure 3.24: Aggregate embeddings of 100 runs for embedding a rhombus shape within the initial Seljuk pattern.

In another experiment, when we aggregate multiple embeddings of a pentagram shape within the extended Seljuk pattern, we obtain two ten-legged star shapes at the center of the weaves (Figure 3.25).


C:\hacer\MATLAB2009_Works\GeneticAlgor

C:\hacer\MATLAB2009_Works\GeneticAlgor

☐ Reference Shape

Estimated Parameters (with GA)
rotation angle: 339.5944
scale: 1.0090
translation: 251.1965 14.27354
generations: 40
☑ Enable Save Mode

Accurrate Alignment (Pattern Search)
rotation angle: -
scale: -
translation: -
iterations: -

rho: 4

Batch Tests
Test Count: 20
Start Index: 0

Display Options
rotation angle: 0
scale: 1
translation: 0 0

☑ Show Multiple Results

Figure 3.25: Aggregate embeddings of 20 runs for embedding a pentagram shape within the initial Seljuk pattern.

These examples show that the algorithm has no difficulty in embedding lines or planar shapes within planes.

3.6.3 Embedding Implicit Parts

It is best to remark here again that in the proposed method, embedding is performed utilizing the weighted canvas instead of the original shapes. As a result of the interaction among the neighboring pixels within the canvas and the GA that utilizes them to locate a best alignment in a run, part embedding algorithm may reveal some implicit parts where originally there is no part drawn explicitly on the canvas.

In an example, the application is able to embed many triangles within a shape (Figure 3.26) where it may be difficult to recognize the triangles in it. However, the application works in such a way that it is as if the missing parts are completed.

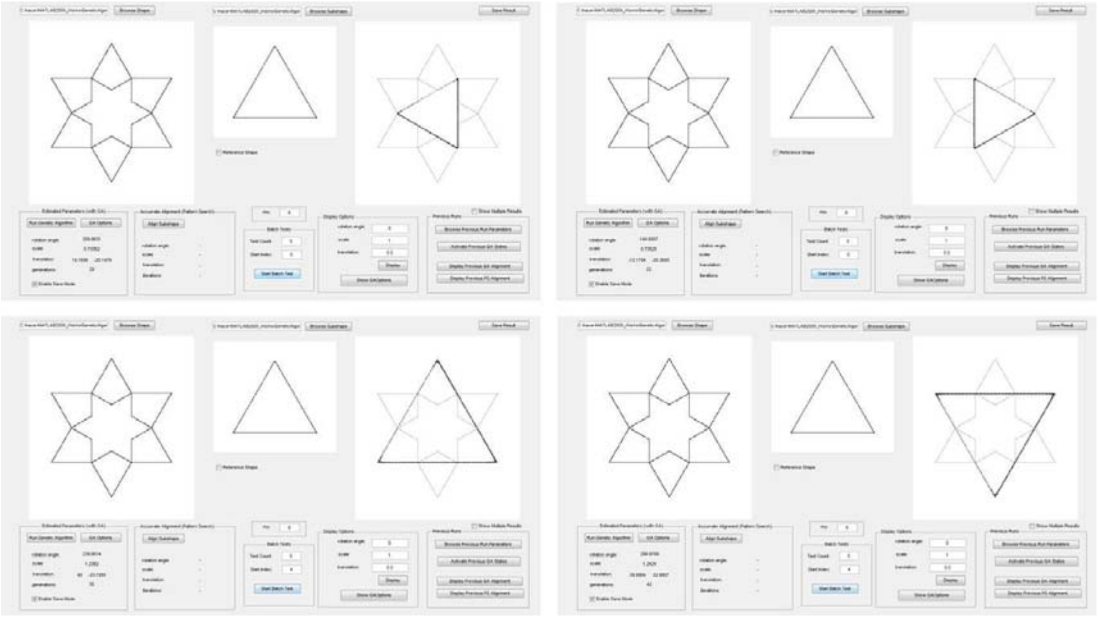
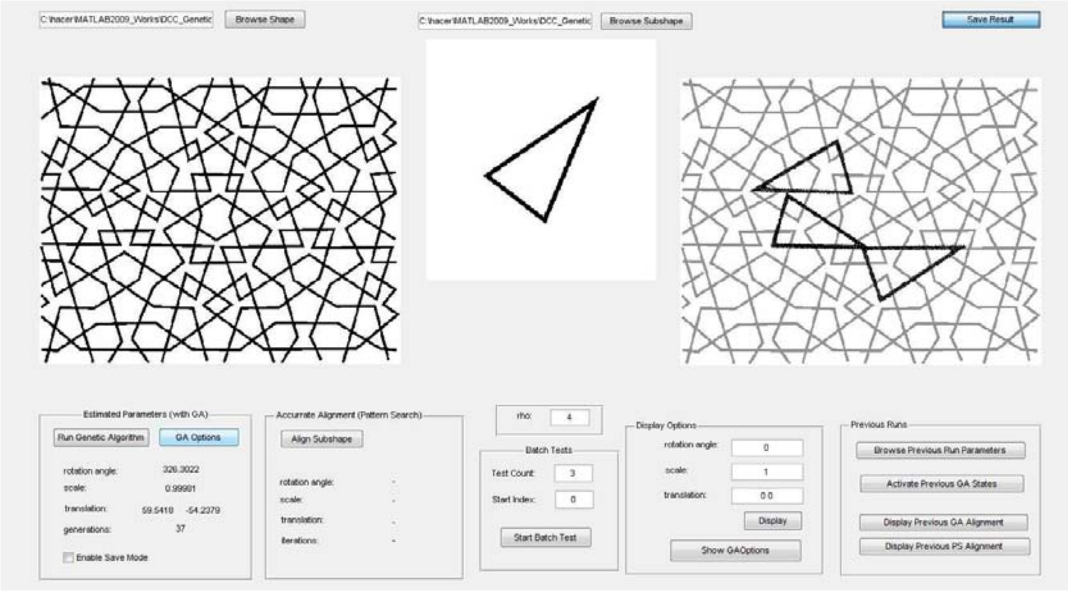


Figure 3.26: Sample implicit part embeddings.

The variety of the embeddings in this Seljuk pattern can be increased by running the algorithm with more simple geometric forms as parts to be detected. For a triangle shape as the part shape, the algorithm embeds various instances of it in the pattern. Three of such embeddings are depicted in Figure 3.27. In all of these embeddings, some of the embedded parts can not be matched to any part of the initial shape on the canvas. These embeddings are similar to the implicit embeddings displayed in the Figures 3.26 in the way that as if the missing parts of the initial shape are completed. We consider this as a potential in creative designs as it implies picking out parts that do not actually exist but belong to perceived wholes. This is consistent with Gestalt rules of visual and spatial perception.



C:\hacer\MATLAB2009_Works\DCC_Genetic

C:\hacer\MATLAB2009_Works\DCC_Genetic

Estimated Parameters (with GA)

rotation angle: 326.3022

scale: 0.99981

translation: 59.5418 -54.2379

generations: 37

☐ Enable Save Mode

Accurrate Alignment (Pattern Search)

rotation angle: -

scale: -

translation: -

iterations: -

rho: 4

Batch Tests

Test Count: 3

Start Index: 0

Display Options

rotation angle: 0

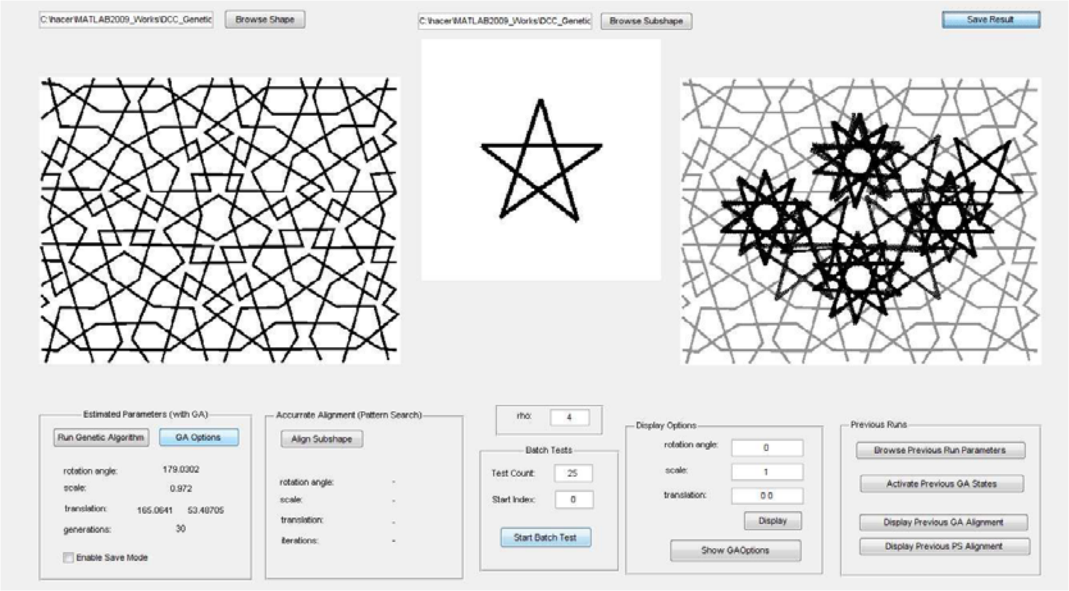
scale: 1

translation: 0 0

Figure 3.27: Sample part embeddings.

We interpret such embeddings provided by the proposed method as an opportunity to realize implicit shapes within a given shape. In other words, it reflects the power behind the proposed algorithm that insists upon searching through the canvas to make best alignment using the given shapes. Such alignments, if not preferred, may easily be avoided simply checking the overlapping parts of the two shapes.

Similarly, iterations of the embedding may help us perceive shapes or shape relations that are not visible the first time around. In the Seljuk pattern considered, the areas between the lines are similar in size and the number of repeating shapes is high. Although the eye groups certain parts at times, any hierarchies in the pattern are not imminent. When we iteratively embed a star shape of a particular size within the pattern, we obtain a cluster of symmetrically located star alignments that are close to one another on the canvas (Figure 3.28). The algorithm matches the star to all of the relevant line clusters in the crowd. When we look back to the original canvas, a configuration of four nodes, which we may not be aware of before, becomes apparent in the center of these implicit embeddings. The star shape acts as a probe for the exploration of the pattern and makes certain implicit geometric relations explicit. Such embeddings within a tile may be visually helpful for the analysis of the line crowded images.


Browse Shape
Browse Subshape
Save Result
Estimated Parameters (with GA)
Run Genetic Algorithm
GA Options
Accurrate Alignment (Pattern Search)
Align Subshape
Batch Tests
Start Batch Test
Display Options
Display
Show GAOptions
Previous Runs
Browse Previous Run Parameters
Activate Previous GA States
Display Previous GA Alignment
Display Previous PS Alignment

C:\hacer\MATLAB2009_Works\DCC_Genetic

C:\hacer\MATLAB2009_Works\DCC_Genetic

Estimated Parameters (with GA)
rotation angle: 179.0302
scale: 0.972
translation: 165.0641 53.48705
generations: 30
☐ Enable Save Mode

Accurrate Alignment (Pattern Search)
rotation angle: -
scale: -
translation: -
iterations: -

rho: 4

Batch Tests
Test Count: 25
Start Index: 0

Display Options
rotation angle: 0
scale: 1
translation: 0 0

Figure 3.28: Sample star-shape embeddings within the tile. The obtained embedding result on the right-most window of the GUI is the result of 25 different GA iterations. GA parameters: number of population=100, scale range=[0.95, 1.1], position range = [-200,+200], orientation range=[0,359], number of generations 30..

In Seljuk geometric pattern analysis, this trait of picking out emerging parts is invaluable in order to decipher which possible visual rules may have been applied in its construction. The example in Figure 3.28 is significant as it helps the eye detect some repeating nodes in the infinite pattern towards understanding the geometric relations underlying it.

As already implied in the sections above, the proposed method is indifferent to shape history or composition and it allows for computing with emerging shapes which we show by embedding a new shape in the product of Figure 3.28. Figure 3.29 shows that the algorithm detects a ten-legged star in the canvas that was previously constructed out of five-legged stars and other lines. Moreover, in correlation with those illustrated in Figure 3.22 above, the algorithm detects line shapes of different widths in the planar shape formed by overlapping stars very much like in design sketches when scribbled lines overlap. This is a good showcase for how the implementation of embedding parts across shape algebras is relevant to computing design sketches where seeing planar marks or new lines in line scribbles is the practice.

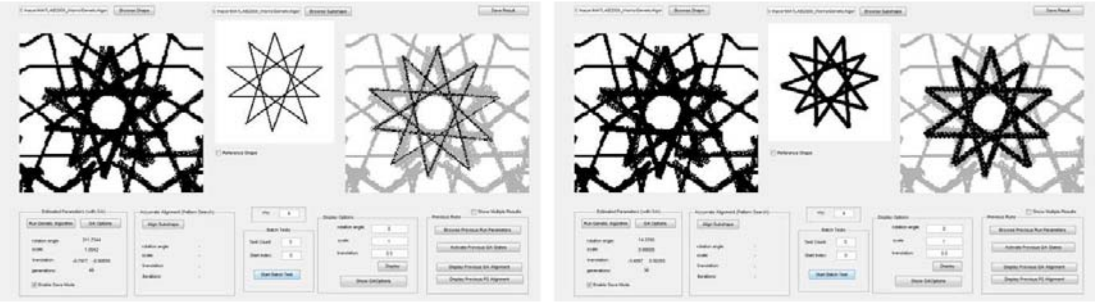


Figure 3.29: (left) Embedding a thin ten-legged star shape of one pixel-width; (right) Embedding a thick ten-legged star shape of six pixel-width. The initial shape is a part of the resultant shape obtained after the experiment shown in Figure 3.28.

3.7 Summary

The part embedding method discussed in this chapter has two main components. The primary component is the representation of the shapes in U_{ij} algebra in the weighted $U_{(i+1)j}$ algebra for the examination of part relations. This way, any external biases for representing shapes are avoided. Weights extracted from a shape code the interaction among the shape parts and implicitly define the necessary constraints for embedding parts. The secondary component is the utilization of an evolutionary approach via a genetic algorithm for computing the possible transformations for aligning shapes. Evolutionary approach is advantageous for these types of problems where there are more than one solutions in the solution space and all are equally important. Starting with the same configuration settings, at each run possibly a different solution can be identified for part embedding. It is this dynamic property that makes GA suitable for solving this problem. With these two components, the proposed method provides a simple but an elegant solution to the part embedding problem.

The proposed approach is in many ways different from the previous approaches. It provides a solution to part relations of shapes staying true to their continuous nature. Moreover, it considers perceived wholes rather than defined categories of shapes such as line, plane, etc. In addition, the weighted representations of shapes in a higher algebra are utilized as registration marks in matching parts rather than external references.

The implementation of this method with the GUI provides necessary tools for experimenting with various type shapes. The experiments with the line drawings such as Seljuk patterns show that, the parts used for embedding can be defined in any desired way. Moreover, the distinction between line shapes and curved shapes are eliminated. The experiments also show that the method is effective for crowded line drawings. For the analysis of such drawings, the method provides some opportunities like implicit embeddings, where the part shape is aligned with the whole in a way completing the missing parts. As a final remark, the method is also effective with drawings simultaneously containing both lines and planes.

CHAPTER 4

CRITICAL POINT DETECTION IN DIGITAL LINE DRAWINGS

In Chapter 2, our first part embedding method is introduced. The method utilizes externally defined constraints in order to map a given shape to a lower level algebra. The labeled shape which is obtained as a result of this mapping is used to obtain temporary representation of the shape via an over-complete shape graph. These constraints are not strictly defined but depend on the user (/designer)'s intentions. Yet, in the absence of external constraints, a generic mapping can be defined in order to obtain default representations for shapes. In the implementation discussed in Chapter 2, this generic mapping is defined according to an analysis method which determines a set of critical points from a shape considering its intrinsic properties. The analysis method developed for this purpose is a novel method which can be used in the domains where critical point analysis is required. The details of this method will be clarified in this chapter.

This chapter is organized as follows. In the first section the problem is introduced together with a summary of the related literature. Following that, the smooth and continuous field that is utilized during the analysis of shapes is introduced. In the next section, the analysis method is introduced in detail. In the last section, some experimental results are discussed for the selected set of shapes.

4.1 Introduction

In computer vision, many researchers have dealt with the problem of locating points of discontinuities which are commonly called as *critical points*. Excluding the works which impose regularity on the shapes of parts ([7]), most of the works, in parallel with the idea stated in the famous work of Attneave ([4]), regularized the partitioning at the part boundaries. Attneave stated that the information on a curve is concentrated at the points of high curvatures. This definition is narrowed in the seminal work of Hoffman and Richards ([24]) by specifying the part boundaries as the points of negative minima of curvature. Since then, most of the research efforts for representing digital curves focused on the computation of curvatures for each point on the curves.

For a continuous curve, the curvature at a point has an exact mathematical definition and is computed considering the data in an arbitrarily small region around the point. However - unlike the continuous curves - the region of the analysis is not well defined in digital/discrete curves and the exact mathematical definition of curvature does not exist. For these reasons, curvature can only be approximated for digital curves. The approximation depends on the size of the analysis region, which is usually referred in the literature as *the region of support*.

Majority of the works which deal with curve partitioning via critical points focused on the development of good approximations to the curvature for digital curves. These works are usually classified into one of the three categories: (1) the ones which compute direct curvature using the spatial coordinates, (2) the ones which estimate curvature after smoothing by filtering techniques (i.e Gaussian filtering), and (3) the ones which estimate curvature using scale space procedures.

Most of the direct curvature estimation based algorithms compute a significance measure for each point p_i along the curve boundary using some mathematical constructions that involve the set of points in the k-neighborhood of the point represented as an ordered set of points as $\{p_{i-k}, p_{i-k+1} ..., p_i, ..., p_{i+k-1}, p_{i+k}\}$, where k is the region of support parameter. Hence direct curvature computations require a previously determined support region size. (An extensive survey on the direct curvature computations can be found in [73]) Unfortunately, a constant predefined region of support does not work properly since there may be features of different sizes on a curve. Some works in the related literature provide algorithms to automate the computation of a local region of support at each point by considering local properties in the vicinity of the point ([73], [74], [13], [55], [41], [54], [3], [77]). These algorithms do not require any input parameters from outside and thus called non-parametric methods.

The alternative to direct curvature computation is smoothing the contour curve by linear filtering before computing the curvature ([3],[55]). Gaussian filtering is the most widely used filter in the critical point detection algorithms. In these methods, determination of the standard deviation, σ, of the Gaussian filter is critical. The value of the σ depends on different factors such as the noise level and scale of relevant features. Since feature size may vary within a curve, multiple values of the standard deviation are often needed to detect features of different scales reliably. In order to solve this problem, the scale space procedures emerged. These methods generate a set of parametrized curves in different scales each obtained with a Gaussian of different standard deviation ([6], [57], [50], [47], [56]).

All of the aforementioned approaches provide solution for simple closed curves. In most of the methods, which compute curvature directly, it is assumed that each point on the curve has exactly two neighboring points. This is the case, since the curves are obtained from the contours of some physical objects. When the curves are not created this way, they are created with certain curve creation algorithms which provide a series of ordered points. These assumptions make the utilization of these methods for the complex line drawings that we handle in this work (such as Seljuk patterns) impossible. Thus, the details of the curvature based approaches mentioned above will be not be discussed in depth here.

In the context of this study, a novel critical point detection method is developed in which any mathematical approximation to the curvature computation is avoided as to determine critical points. It is based on the analysis of the deviation of a curve from a straight line.

The content of this chapter is prepared to introduce the novel approach that is suitable for detecting critical regions in complex digital line drawings. After the critical regions are determined, finding the critical points within these regions is the easy part of this problem and can be implemented specific to the needs of the application. The proposed method for critical region detection works efficiently for any type of planar curves, including the ones with discontinuities or self-intersections. The novelty is in that during the computations, any geometrical notion related with the images is avoided. Instead, the intrinsic properties of planar curves are analyzed using a smooth and continuous field extracted from the curve images. This field, which codes the interactions among the pixels, is analyzed using ellipse shaped kernels. The statistical analysis of the field within a kernel window gives the distinguished/critical regions when a proper reference kernel, which is actually a kernel belonging to a straight line, is taken into consideration.

The proposed approach provides a fully automated iterative mechanism to determine the set of critical regions within line images. For each determined critical region, the analysis is repeated at different resolutions, depending on the properties of the region. Starting from a low resolution analysis, the approach increases the analysis resolution at each iteration step considering the field encapsulated within the regions.

Throughout the chapter, a set of line drawings each with different intrinsic properties will be utilized to discuss the details of the proposed method. These shapes are depicted below. In (a), the shape is a composition of a curve with a highly free-form character and a straight line, while (b) is a composition of five perfect circles and (c) is a composition of straight line segments. Each of these three samples may be considered as a representive for a class of similar shapes.

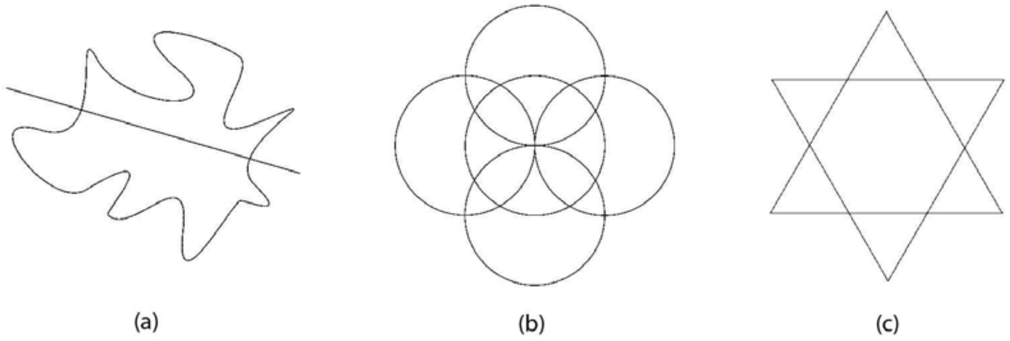


Figure 4.1: Selected set of shapes.

4.2 The Edge Strength Function ([71])

The edge strength function, w, which we have utilized as the weight function in Chapter 3-equation (3.1), provides a smooth and continuous field by modeling the interaction among the boundary pixels of an image. This function is crucial for the proposed analysis method, because during the analysis, the criticality of a pixel is judged considering the interaction of the pixel with its surroundings. The field w_p at a well defined local neighborhood of a pixel p provides the information related with the intrinsic properties of that region. What we mean by a well defined local neighborhood of a pixel will be clarified in the upcoming sections.

The edge strength functions corresponding to the sample shapes in Figure 4.1 are as shown below.

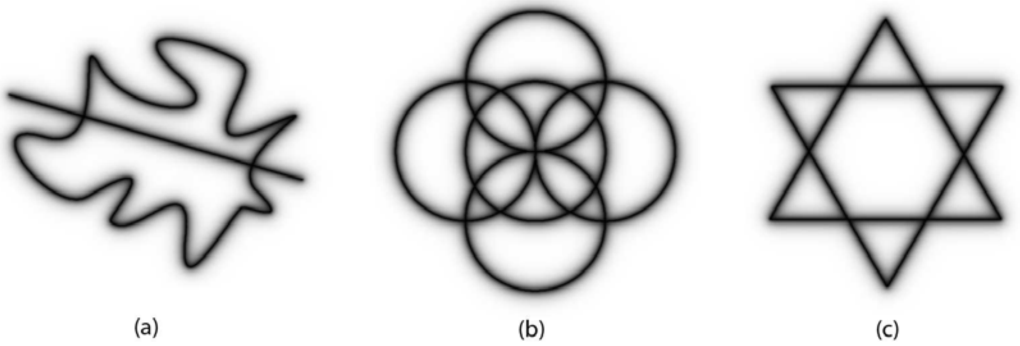


Figure 4.2: Corresponding edge strength functions of the shapes in Figure 4.1.

4.3 Detection of Critical Regions

The critical region detection method is based on a simple idea: a point is an ordinary point if it is located in a neighborhood in which shape pattern in that neighborhood does not deviate much from a straight line. According to this idea, the characteristic properties of a close neighborhood of a point are used to determine the criticality of the point. In the proposed approach, the analysis is based on the field occurred around a point, which implicitly models the interaction of the pixel with its local neighborhood. In this section the details of the analysis of this field is explained in order to decide the critical regions within a shape.

4.3.1 Ellipse Shaped Analysis Windows

The analysis of the local properties of a point is carried out considering the properties of the continuous field in a close neighborhood of that point. This is achieved by analyzing the field using ellipse shaped analysis windows (or elliptic windows in short) which are aligned through the flow of the field considering the gradient values. By means of the elliptic windows the local neighborhood of the point in both sides of the line pattern is symmetrically evaluated in equal weights. Some sample window alignments are depicted in Figure 4.3.

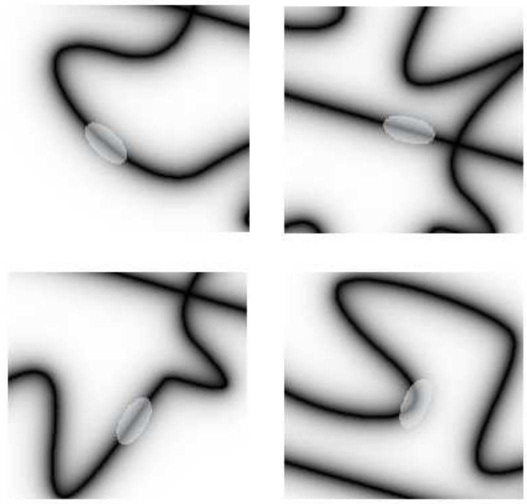


Figure 4.3: Sample ellipse window alignments

During the analysis, for each boundary point of a shape, elliptic windows are aligned considering the gradient direction at that point. The gradient direction and the ellipse alignment direction are displayed in Figure 4.4.

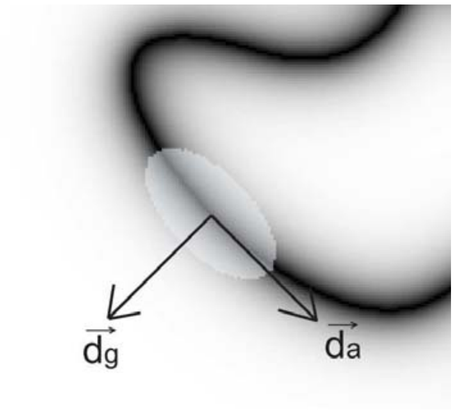


Figure 4.4: Gradient direction (d_g) and the ellipse alignment direction (d_a) for a sample ellipse window.

The details of how we utilize these windows as features in our analysis will be made clear after the discussion related with the reference analysis windows.

4.3.2 Reference Analysis Windows

In order to design an automated system, there is a need to find an independent evaluation mechanism for judging the feature values at the analysis windows. This can be achieved based on the idea that the deviation of the field around a point from that of a straight line determines the criticality of the point. Accordingly, an infinite length straight line can be utilized as the reference shape and the reference window values can be computed from the field induced from the reference shape. An infinite length straight line is the proper choice in this approach since the curvature can be defined intuitively as the amount of deviation from being flat/straight in the case of a line (this is in accordance with Attneave's observation ([4]) that critical information on a curve is located at the areas of high curvatures).

The proposed method determines proper ellipse dimensions at each analyzed point by considering the window mean values for that point for a range of ellipse dimensions. In order to judge the criticality of the points based on the window mean values, a range of reference window mean values are pre-computed with changing ellipse dimensions over the field induced from an infinite length straight line. An arbitrary point on this line is selected as the center for the ellipse windows (Figure 4.5).

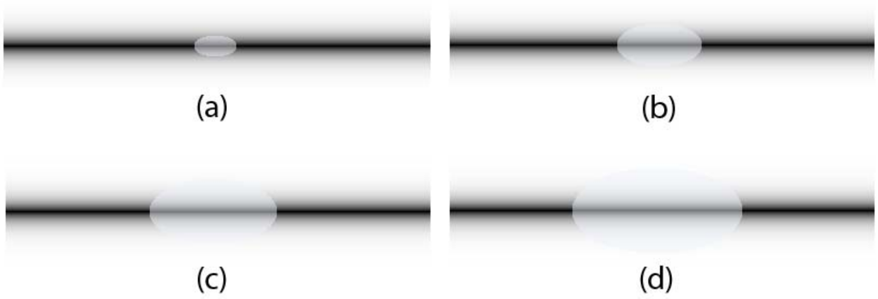


Figure 4.5: The reference shape and ellipse windows in different dimensions (KMV: window mean value): (a) ellipse minor ax. dim.: 15 px; major ax. dim: 30 px; KMV: 0.5087, (b) minor a.d.: 20 px; major a.d.: 40 px; KMV: 0.4239, (c) minor a.d.: 30 px; major a.d.: 60 px; KMV: 0.3121, (d) minor a.d.: 40 px; major a.d.: 80 px, KMV: 0.2433.

For a range of different ellipse windows which are centered at the selected reference point, the reference window mean distribution is computed. This distribution is utilized as a reference in order to evaluate the criticality of a point for a range of window dimensions (Figure 4.6).

The axis labeled as *window size* in the depicted distribution corresponds to the minor axis dimension of the ellipse windows. We set the major axis dimension in our experiments to two times the size of the minor axis in this implementation. Although this is not a strict requirement, for the experiments provided in this chapter, this setting works well.

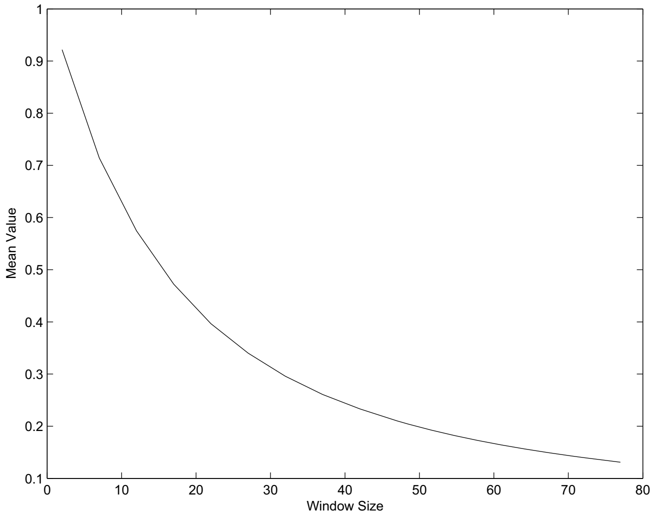


Figure 4.6: Distribution of the Reference Window Mean Values.

As the ellipse dimensions become larger the window mean values decreases. This is expected when the field around the reference shape is considered. The field of the reference shape is computed by setting the smoothness parameter, ρ, to 8. For different ρ values, the reference window mean distribution values would change, although the character of the distribution would be similar. The change in smoothness parameter affects the scale of the analysis, hence the accuracy of the analysis changes. For a specific shape, if we increase ρ while computing the fields - for both the analyzed shape and the reference shape - some of the previously determined critical regions might fade away depending on the amount of increase in smoothness. This effect is analogous to increasing the dimension of support regions in discrete curvature computations.

It is important to compute the fields for both the analyzed shape and the reference shape with the same diffusion parameters. Hence, if the scale of the analysis is to be changed, the reference window mean distribution should be adapted accordingly.

4.3.3 Critical Regions

The aim of detecting the critical regions is to find the set of regions in the image that contain structurally important parts which can be used to represent a shape. The analysis is performed over the field induced from the image boundaries. Depending on the intrinsic properties of the regions in the image, the distribution of the field will differ. In order to capture critical regions, a two phase analysis is made. In the first phase, a local evaluation of the field around the shape points is performed with the elliptic windows. In the second phase, the identified regions in the first phase are merged according to their proximities and a merged region analysis is performed considering the field distribution in the merged regions.

The most important part of this analysis is determining the true size of the ellipses for the identification of the regions with different intrinsic properties. Consider the four sample regions depicted in Figure 4.7. In the analysis of this shape, a predefined and fixed ellipse dimension can not capture all these regions. For example, if the ellipse dimension were set to a proper size to capture the region labeled as (d), the region around label (b) would be missed.

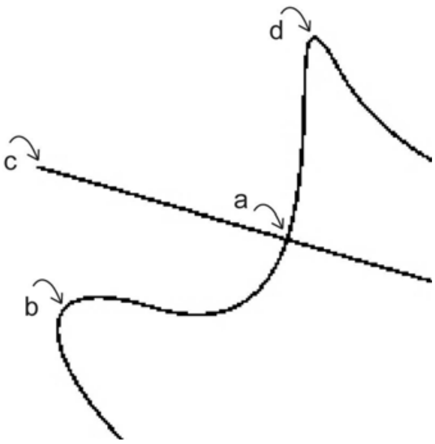


Figure 4.7: Sample critical regions with different intrinsic properties.

In fact, with no previous knowledge about the shape, the proper size of the ellipse for the region labeled as (b) is not easy to guess. It is due to the reason that, the feature around that region is not as sharp as the one around the region labeled as (d). In order to avoid predetermination of the ellipse dimensions and capture varying size features in the same image, an adaptive method is necessary. For this reason, an iterative method is developed to adapt the ellipse dimensions considering the field properties of distinct regions within a shape.

4.3.3.1 Region Analysis: First Phase

The analysis starts from the coarsest level in which the ellipse dimension is set to its maximum allowed sizes for each and every image pixel/point. This maximum size is determined in proportion with the bounding box of the shape within the image. The dimension of the major axis for the ellipse is computed as half of the diagonal line of the shape bounding box in the implementation used in all the experiments provided in this section. The ellipse in this big proportion is selected to initiate a conservative analysis.

Depending on the smoothness parameter (ρ) of the field, the effective ellipse window dimensions are implicitly defined. When the ellipse dimension is lower than the effective minimum size, the mean field value within the window will tend to be close to 1. On the other hand, if the ellipse dimension is greater than the effective maximum size, the mean field value will tend to be close to 0. In order to illustrate this, let us re-consider the reference window mean distribution shown in Figure 4.6 for *rho* set to 8. A limited range of ellipse dimensions can be effective for evaluating the windows during the analysis in this distribution. The effective range is depicted in Figure 4.8 with a rectangular region.

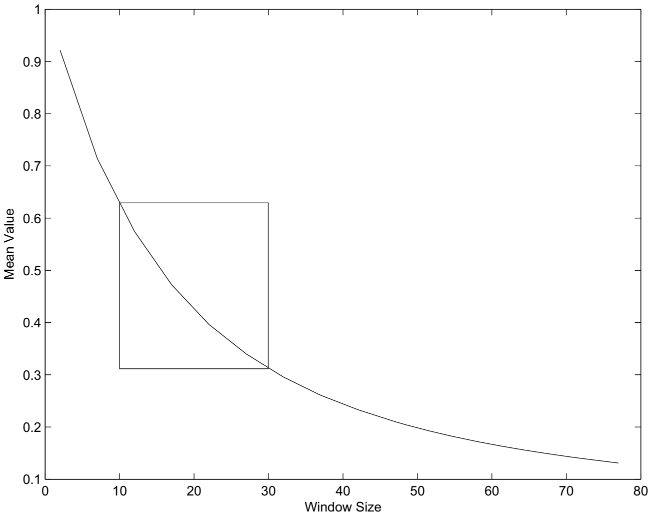


Figure 4.8: Effective range of ellipse axis dimensions and the reference window mean distribution, which is computed with *rho*=8.

When the window size is less than 10 pixels, the mean value within the reference window, which goes above 0.6, is too high to discriminate the high intensity regions from the average intensity regions. Similarly, when the window size is greater than 30 pixels, the intensity of the field encapsulated within the reference window reduces below 0.3. Above this dimension resolution of the analysis reduces and the field becomes no different than a line. Therefore, at its coarsest level the maximum allowed ellipse dimension for the minor axis is 30 pixels for this reference mean distribution. In other words, if the bounding box analysis results in an ellipse dimension more than this allowed size, it is reset as the allowed maximum size.

Let us restate here that, the smoothness parameter of the edge strength function determines the resolution of the analysis by implicitly influencing the effective window sizes for the mean reference window distribution. As a result of that, the intrinsic properties of the identified regions change. In all the experiments we provided in this chapter, the smoothness parameter is set to 8.

At the start of the analysis, each of the points on the shape boundary is treated as a candidate critical point. For each ellipse window alignment at a point, the mean value of the field encapsulated within the window is stored as the feature of that point. The computed window mean values at varying ellipse dimensions are interpreted as a sign of criticality depending on the deviation from the reference window mean values of the corresponding ellipse dimensions. In the ideal case *if the deviation from the reference window mean value is close to zero, the point is considered as an ordinary point, hence not critical.* If the deviation of the window mean of the point from the reference window value is higher than some predetermined degree, than the point is marked as a candidate for being critical.

Assume that M_p^e is the mean window value for the ellipse dimension e at the analyzed point p, and R^e is the reference window mean value for the same ellipse dimension e. Assume that N_p^e is the normalized window mean value of the point with the reference window mean value as:

$$N_p^e = \frac{M_p^e}{R^e}$$

,

the function f_c which determines the criticality of a point can be defined as:

$$f_c = \begin{cases} 1 & \text{if } N_p^e > 0.9 \text{ and } N_p^e < 1.1 \\ 0 & \text{otherwise} \end{cases}$$

As its definition implies, the function f_c returns 1 for a point which is assumed to be critical and 0 otherwise. The predefined parameters [0.9,1.1] works conservatively as we desired for our experiments. In order to detect only sharp features for example, this range should be reduced vice versa.

Each region depicted with circles in Figure 4.7, contain many points marked as critical points due to the fact that the deviation of the field in the neighborhood of the points in that region is high from that of the reference shape. Hence, a set of points appears as critical in the proximity of these regions. The non-uniform interactions of the points within these regions cause strong deviation of the fields from that of the straight lines. After determining the set of candidate critical points in an image, a second phase of evaluation is required around the neighboring regions of these points to increase the accuracy of the critical region analysis in each iteration. The aim is *reducing the dimensions of the candidate critical regions until a minimum region is obtained which still deviates sufficiently from the reference/straight line.*

Assume that P_c is the set of points that are marked as critical in the first step of the evaluation. For each point p in P_c, a circular disk of radius equal to the minor axis of the ellipse dimension, which is used to identify that point, is put around that point. This is performed for each point in P_c so that image is partitioned into two types of regions as critical or not (Figure 4.9-a). The overlapping critical regions (including the touching regions) in this partitioned image is merged to obtain one well defined region. Merge operation is designed to consider the total region area enclosed in these regions and the weighted centroid of the field within the regions. A disc whose area is equivalent to the enclosed area of the overlapping/connected regions is placed to the computed centroid of those regions (Figure 4.9-b). This region merging is necessary to determine the most suitable ellipse dimension for each region in the next iteration step.

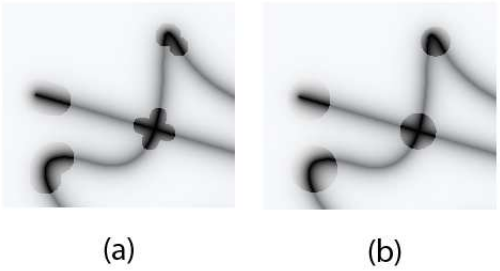


Figure 4.9: (a) Critical regions, (b) Merged critical regions.

4.3.3.2 Region Analysis: Second Phase

In the first phase of the analysis, a set of candidate critical regions are obtained by merging connected regions to obtain one region that statistically summarizes the contents of those merged regions. Each merged region is represented with a window mean value obtained from the field encapsulated within the merged region. This value is a key to evaluate the criticality of that region and to determine the proper size of the ellipses for the points in that region for the next iteration. Let us restate here that the analysis goes from coarser to finer so that at each iteration the analysis window sizes are reduced considering the distribution of the fields within the considered regions.

In this phase, once again the same reference shape (infinite length straight line) is used as the reference for evaluation of the the merged critical regions. This time, instead of the elliptic windows, disc-shaped windows are used to extract the reference distributions from the field induced by the reference shape for varying dimensions of discs (Figure 4.10).

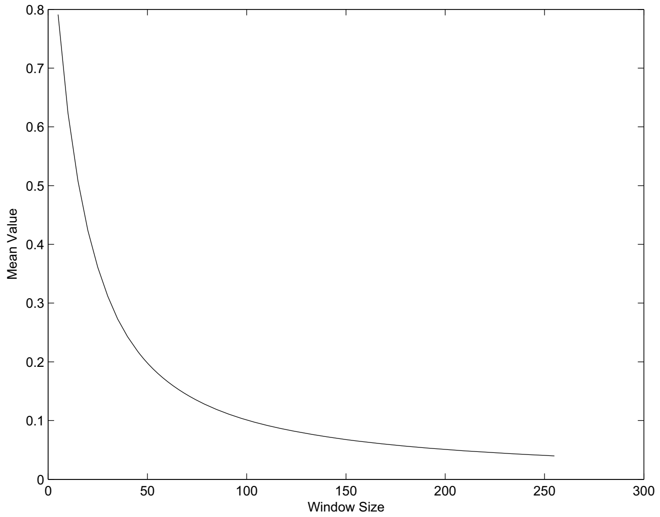


Figure 4.10: Distribution of the field mean values over varying disc dimensions.

Assume that M_r^d is the mean value of the disc with radius d for the region r, and R^d is the reference disc mean value for the same dimension d. For each merged critical region, r, the function g in (4.1) gives the scale coefficient for that particular region that will be used to redefine the ellipse dimension for that region for the next iteration.

$$g(D) = e^{-D/2} \tag{4.1}$$

Where D is the normalized deviation of the region disc mean value from the reference disc mean value:

$$D = \frac{\left|M_r^d - R^d\right|}{R^d}$$

The higher the normalized disc deviation within a region the faster the reduction of the ellipse dimension according to the scale coefficient returned by the function g. As the disc mean value approaches closer to the reference mean value, the reduction in the ellipse dimension is lessened.

4.3.4 Convergence Criteria

The aim of each iteration in the proposed method is to define the boundaries of critical regions as accurate as possible. In that sense, from a coarse to fine resolution analysis is performed via ellipse shaped windows and disc-shaped windows in a two phase analysis. The second phase of the analysis determines the proper ellipse dimension for each region for the next iteration. The reduction rate of the ellipse dimension is lessened as the deviation from the reference disc mean value is lessened. The reduction in the ellipse dimensions however has a lower limit due to the reasons we discussed in Section 4.3.3.1. The reduction of the ellipse dimension for a region is stopped when the ellipse dimension reached to a value less than or equal to the minimum allowed window size. For the particular case, in which *rho* is 8, the minimum allowed window size is 10 for example. If a region is still a candidate even for that small ellipse dimension, it is accepted as the critical region.

Another stopping condition is based on the degree of deviations in successive steps. The key idea behind this criteria can be stated as follows. If after the first phase of a coarser analysis a region is marked as a candidate region, there is at least one critical region that causes the high deviation to make this coarser level region a candidate. Assume that there is no critical region within the region that is determined as a candidate in the first phase. In that case, the deviation of the mean value of this region should be very close to that of the reference shape (assuming that the analysis window sizes obey to the predefined minimum and maximum constraints). In that case, however, the deviation of this region would be small and would not be marked as a candidate region. However, it is marked as a candidate, so this assumption results in a contradiction, which has a meaning that the candidate region is composed of at least one sub-region that has a field distribution which causes the high deviation from the reference shape.

According to this assumption the second stopping criteria can be explained as follows. For some regions, as the analysis window size is reduced, the deviation degree of the region gets smaller and smaller. At some point, if the ellipse dimension keeps reducing, the critical region disappears. However, according to the assumption we stated above there must be a critical region which disappears for a smaller analysis window (as the green region depicted in Figure 4.7). In order not to miss such regions, the reduction of the window size is stopped when the mean value within the disc approaches very close to the reference value, i.e. the deviation approaches to zero. The method is designed not to miss any regions that are marked as critical regions in the first phase.

The iterations are halted when the critical regions determined by two successive iterations do not change any more. The algorithm is guaranteed to converge since at each step the ellipse dimensions are either reduced or stabilized. For the stabilized ellipse dimensions, the field discs generated in the successive iteration steps do not change. The discs are continuously shrink in each iteration due to the increased accuracy of the boundaries of critical regions. At a point, since the ellipse dimensions have a lower limit in size, the discs are guaranteed to be stabilized. As a result, after some finite number of iterations, the algorithm successfully produces the set of critical regions.

4.4 Critical Point Detection

After the critical regions are detected (as explained in Section 4.3), the critical points within these regions can be detected in various ways, depending on the accuracy requirements of an application.

A pixel based analysis within these regions provides us the critical points of interest. In the context of this work, critical regions are examined to determine the points/pixels located at the topologically important points. These points are utilized to structurally represent shapes with the over-complete shape graphs. The pixel based analysis marks the pixels that are located at the discontinuities as critical pixels. Those pixels are the ones that are either located at junctions, at corners or at the boundaries, which can easily be determined by structural image analysis methods. For some regions, none of these types of the pixels exist. Those regions are composed of pixels which form a smooth and continuous line some degree of bending. For those regions, the closest pixel to the weighted field centroid of the region is assumed to be critical in our analysis.

4.5 Experiments and Discussions

This section provides the results of the critical region detection method for the set of sample shapes that are depicted in Figure 4.11. For each sample shape, the candidate regions determined after some iterations are displayed.

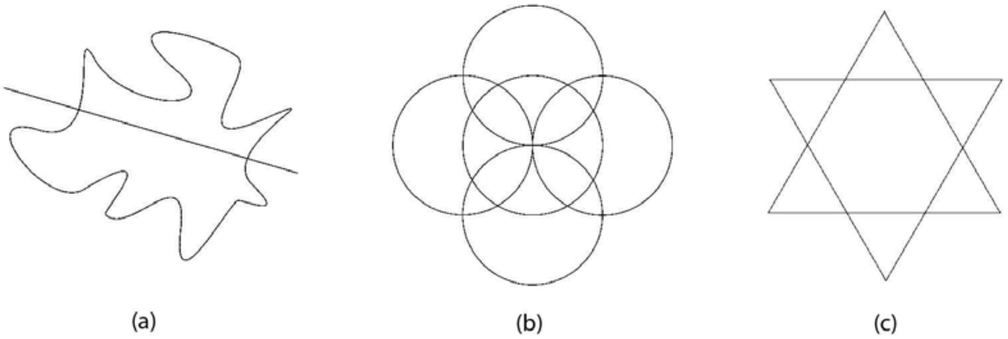


Figure 4.11: Sample shapes for the experiments.

The first sample shape is composed of a closed curve in a highly free-form nature and a straight line passing through that curve (Figure 4.11-(a)). The image size is 587x491 pixels. The size of the bounding box of this image is very large so that the initial ellipse dimension is automatically set to the maximum allowed size: 30 pixels (for the field with *rho* = 8). As the number of iterations increases, the analysis resolution increases automatically according to the method that is explained in the first phase of the analysis. The set of candidate critical regions as a result of the analysis is depicted in Figure 4.12.

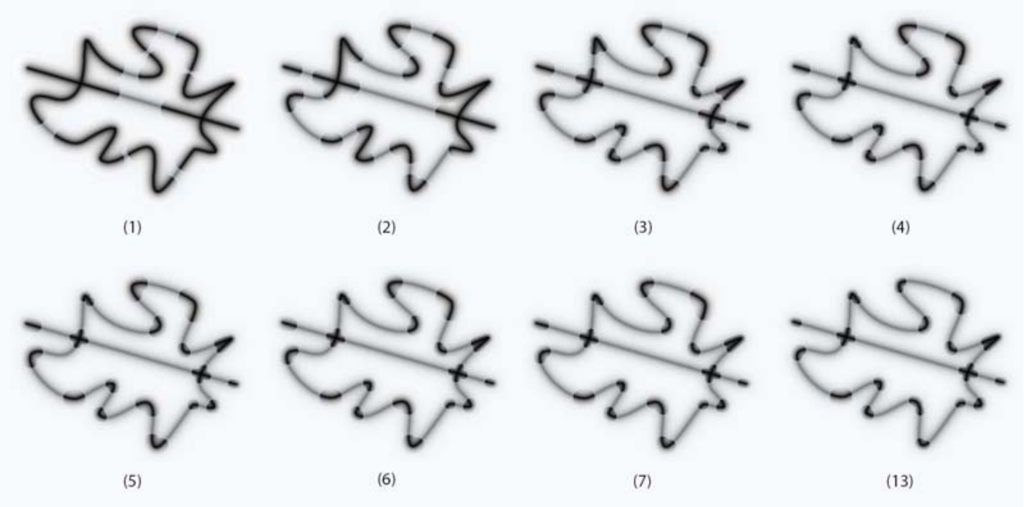


Figure 4.12: The candidate critical regions, shown in bold lines, are determined after the first phase of the analysis. The number below each shape stands for the iteration index.

After only a few iterations, the analysis window decreases considerably fast. For the sample shape shown above, after the fourth iteration, the decrease in the dimensions of the critical regions slows down. This is due to the reason that the analysis window size is decreased according to the exponential function parameterized by the deviation from the reference window. Deviation at each region decreases, hence accordingly, the rate of change in the critical region dimension decreases as the analysis evolves. The corresponding merged regions that are obtained in the second phase of the analysis are depicted in Figure 4.13.

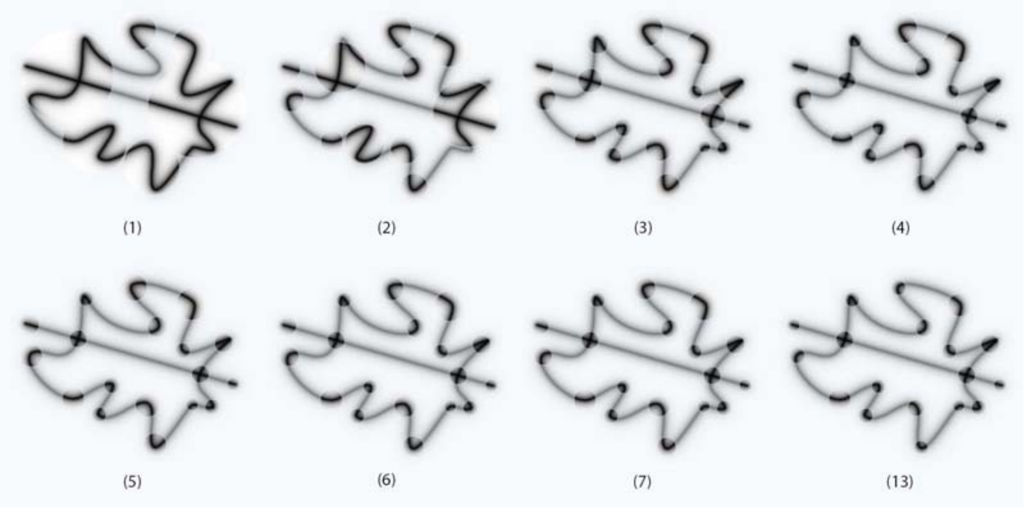


Figure 4.13: The merged critical regions obtained after the second phase of the analysis. The numbers stand for the iteration indices. The result depicted as the 13'th iteration is the converged result.

The regions determined in the intermediate iterations show that, the accuracy of the analysis increases at each successive step. It is visible from the marked regions that the dimensions of the critical regions and their evolution speed change for the regions according to their intrinsic properties.

The second sample shape is composed of five intersecting circles (Figure 4.11-(b)). The image dimension for the corresponding sample image is 539x538 pixels. The critical regions as a result of the proposed analysis method after the first phase is depicted for some selected iterations in Figure 4.14.

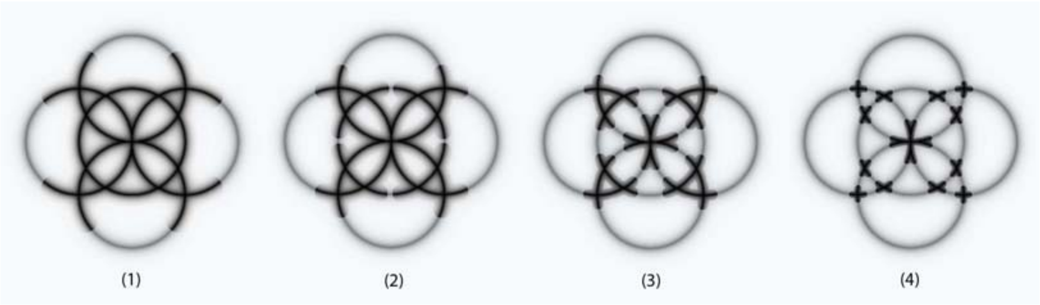


Figure 4.14: The candidate critical regions determined after the first phase of the analysis. This sample is converged after 4 iterations.

In the first iteration, the ellipse dimension is proportional to the bounding box of the image. As a result of that many points in the image are marked as critical points. Therefore, the first iteration results in a very large disc that approximately includes the whole image (Figure 4.15-(1)). However, since the deviation of this disc from the reference shape is high, its size is reduced very fast to converge in three more iteration steps. The resultant critical region is shown in Figure 4.15-(4).

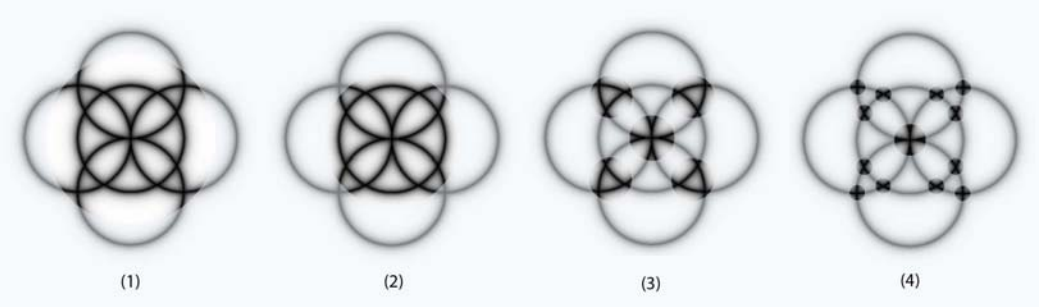


Figure 4.15: The merged critical regions obtained after the second phase of the analysis.

The third sample shape is composed of only straight lines (Figure 4.11-(c)). The regions determined after the first phase are displayed in Figure 4.16. The dimension of the corresponding image is 471x523 pixels.

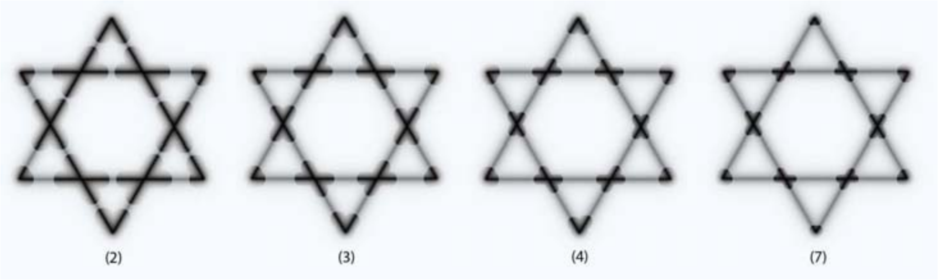


Figure 4.16: The candidate critical regions determined after the first phase of the analysis. This sample is converged after 7 iterations.

The merged critical regions obtained after the second phase is depicted in Figure 4.17.

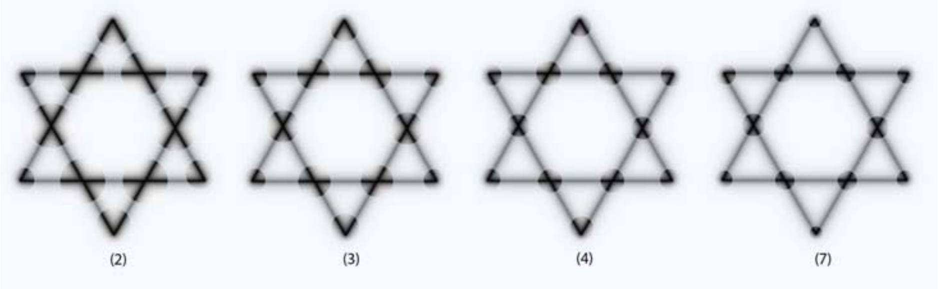


Figure 4.17: The merged critical regions obtained in the second phase of the analysis.

The sample shapes are selected in such a way to include lines with different intrinsic properties. In that respect, the first sample is a hybrid shape that is composed of curved and straight line segments; the second sample is composed of curved lines and the last sample is composed of straight lines only. The results of the analysis for these data show that the proposed method is effective to detect structurally critical regions in line images with different characteristics.

In the proposed method, data is not processed considering the pixel coordinate values as it is in the direct curvature based methods. This approach considers the field distribution within a well defined window which makes it robust to discretization noise. Moreover, changing the reference shape of the analysis helps extraction of the critical regions from the shapes which are created using different brush types. For example, a brush type whose lines highly deviate from straight lines can be used to construct shapes. A sample reference shape that is constructed with such a brush is depicted in Figure 4.18).

Figure 4.18: A different reference shape.

The original shape in Figure 4.11-(b) is modified using the same brush with Photoshop. The obtained image is shown in Figure 4.19. It is visible in the magnified window on the left that this modified version of the shape is quite challenging to analyze since the pixels are sprayed all around the line that appears in the original drawing, like pixel clouds in a noisy manner. Although, human vision can perceive the whole from these cluttered particles/pixels, it is very difficult for the computer implementations that make pixel based analysis.

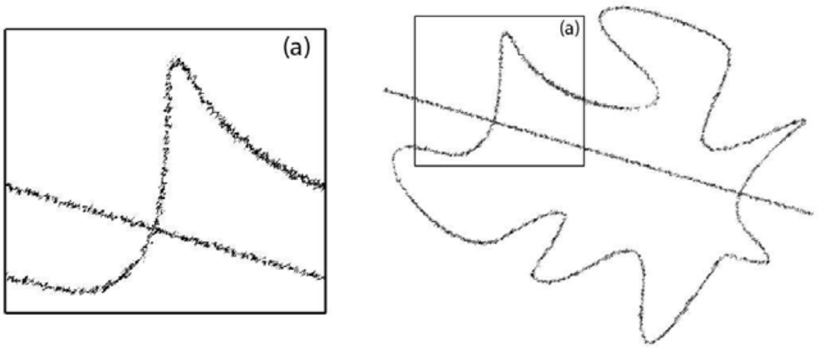


Figure 4.19: The shape in Figure 4.1-(a) is redrawn with the same brush that is used in constructing the reference shape in Figure 4.18.

The proposed approach has no difficulty in determining the critical regions. The regions marked by the method in the first few iterations prove this (Figure 4.20).

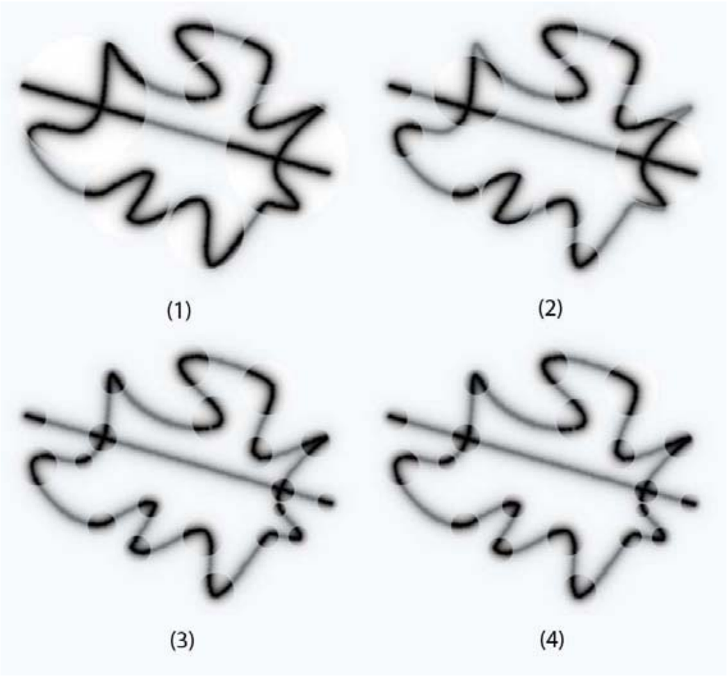


Figure 4.20: The first 4 iterations of the analysis for the shape in Figure 4.19.

The result obtained after the final iteration is shown in Figure 4.21. Marked critical regions, which are obtained for this cluttered drawing, is consistent with the ones obtained from the original drawing (Figure 4.13). The image size in this experiment is 1252x1031, which is approximately two times larger than the original image. Although the scale is doubled, the marked critical regions is still coherent with the original drawing. This is achieved by setting the diffusivity parameter to a larger value. In this experiment ρ is set to 16.

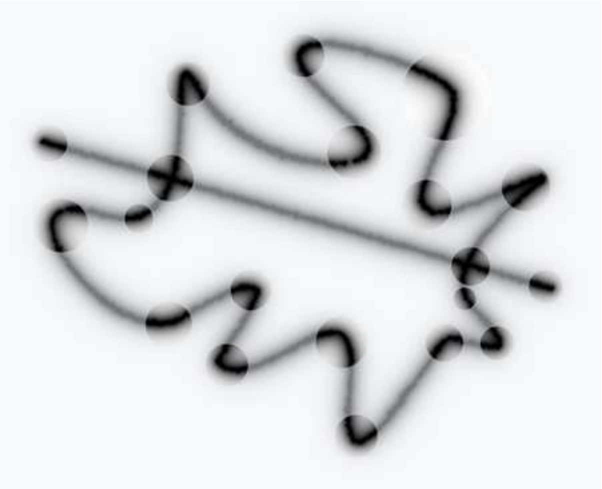


Figure 4.21: Critical regions determined for the cluttered image shown in Figure 4.20.

The approach is effective not only in line drawings but also in the contours obtained from the real image objects. The critical regions of the two sample shapes, fish and bear, obtained from the snapshots of the digital copy of the paper of Latecki and Lakamper [40] are displayed in Figure 4.22 and 4.23. Although, the contour of the fish is very noisy, the marked regions are located very close to the structurally critical regions of the shape.

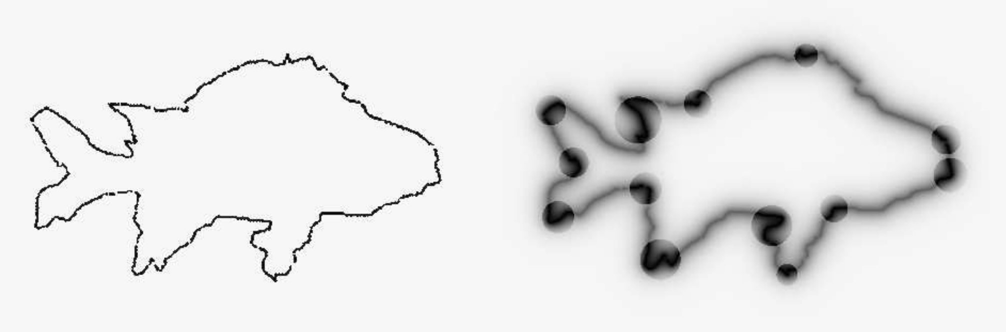


Figure 4.22: Critical regions depicted with circles. The field is computed with (ρ=8).

Similarly, for the bear data (Figure 4.23), the key regions are marked in the regions that correspond to some body parts, like edge of the legs, arms, neck etc. These results show that the method may have a potential usage in representing the object contours for shape analysis and matching applications. Being out of the scope of this work, as a future direction, the infrastructure developed in this research may be adapted to pattern classification and shape analysis applications.

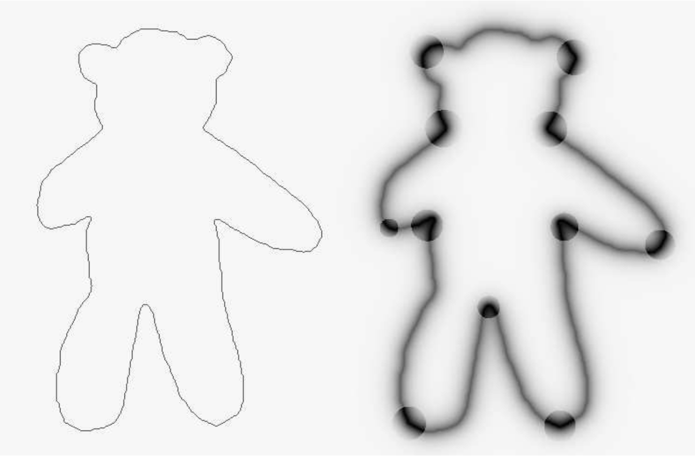


Figure 4.23: Critical regions depicted with circles. The field is computed with (ρ=8).

In addition to these experiments, another set of experiments are performed with the famous three data of Teh and Chin ([73]). These data set is used in many curvature based dominant point detection works to asses the performance of their methods. Most of these methods aim to approximate the curves with polygons. Although, the method proposed in this chapter has no intention of polygonal approximation, the set of the detected critical regions of each curve are depicted below for the sake of completeness.

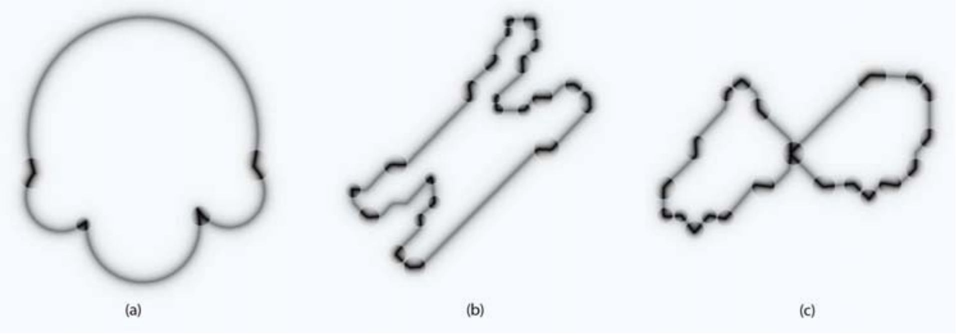


Figure 4.24: Data set created similar to the ones in ([73]). (a) Semicircle-shaped curve, (b) Chromosome-shaped curve, (c) Infinity-shaped curve.

It should be stated here that, the data set is not the exact data that appear in [73]. Except for the semicircle-shaped curve, they are created from the snapshot images of the digital copies obtained from that paper. The semicircle-shaped curve (Figure 4.24-(a)) is reconstructed with parts of high resolution circles in proportion with the original shape. The pixel based applications process the data pixel by pixel, hence the position of each pixel, hence the size of the images, is critical for assessing their methods. On the other hand, the proposed method is not pixel based, and what is important for the method discussed here is not the positions of the individual pixels but the interactions among them. Hence, the reconstruction of this data set does not influence what we discuss in the context of this Chapter.

The semicircle-shaped curve used in our experiments is very smooth and continuous compared to the original one in [73]. There are four regions that violates the continuous nature of this shape. The proposed method identifies all these regions as critical (Figure 4.24-(a)). Other points that lie along the circular arcs are evaluated as uninteresting.

As for the chromosome-shaped data (b) and the infinity-shaped data (c), both has many discontinuities. The identified critical regions in (b) and (c) is comparable with the curvature based methods discussed in [73]. For example, if we approximate the chromosome-shaped data with straight lines utilizing the center of the critical regions as the critical points, we obtain the shape in Figure 4.25.

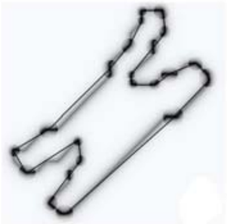


Figure 4.25: Approximation of the chromosome data with the proposed method.

This approximation is comparable with the the results discussed in [73] (Figure 4.26).

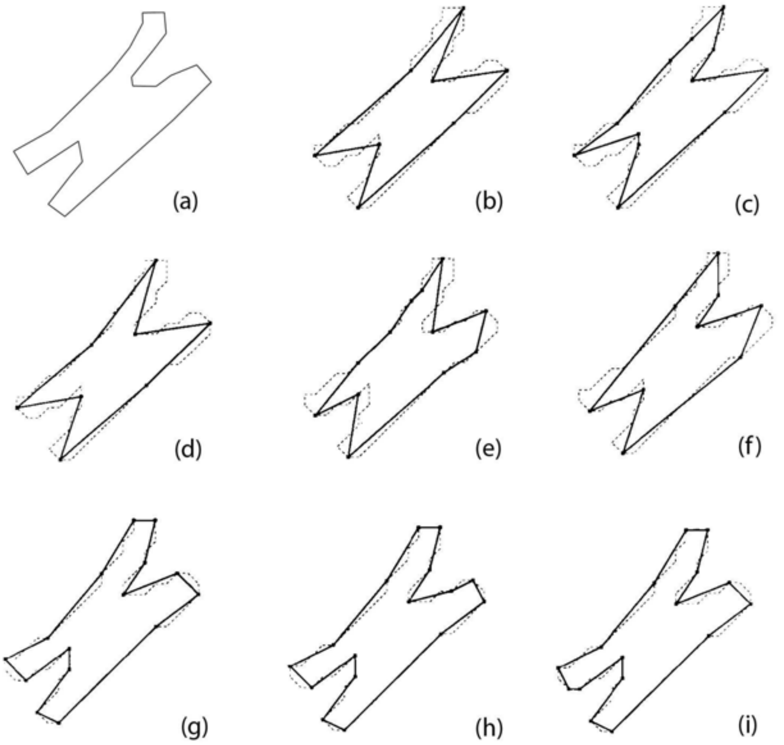


Figure 4.26: (a) The proposed method, (b) Rosenfeld-Johnston algorithm, (c) Rosenfeld-Weszka algorithm, (d) Freeman-Davis algorithm, (e) Sankar-Sharma algorithm, (f) Anderson-Bezdek algorithm, (g) Teh-Chin algorithm (k-cosine), (h) Teh-Chin algorithm (k-curvature),(i) Tel-chin algorithm (1-curvature). All the sample results from (b) to (i) are copied directly from [73].

We used the center of each critical region as the critical points that are used to polygonally approximate the chromosome data. The approximation results for the chromosome-shaped curve that belong to the other eight methods are taken directly from the digital copy of the paper [73] in order to visually compare the approximations[1]. Although, in our method, no mathematical approximation is implemented to find high curvature points, the result we obtained - shown in (a)- is visually similar to the results that belong to well known Teh-Chin algorithms, depicted in (g),(h) and (i). In all these three implementations, first the region of support for each pixels are computed, following that one of the three different approximations to the curvature is implemented.

---

[1] Let us restate again here that, the method proposed in this work has no intension to polygonally approximate the given shapes. The data set we used in these experiments are visually similar to the original digital curves, but not exactly the same. Therefore, comparing the approximation errors numerically does not make sense in the context of this discussion.

In their pioneering paper [73], Teh and Chin argue that *the detection of dominant points relies not only on the accuracy of the measures of significance, but primarily on the precise determination of the region of support.* The results we depicted in Figure 4.26 actually support this argument. It is due to the fact that after the computation of critical regions, we do nothing but select the center of these regions as the critical points to obtain the approximation shown in (a). No measure of significance is computed at all.

4.6 Summary

In this Chapter, a novel approach to the critical region detection problem is introduced. The novelty of this approach is in its measurement of the significance of points where instead of pixel based analysis and using mathematical abstractions for widely accepted significance measure as curvature; a field based analysis is performed. This measure provides a number of advantages; it is (1) closely related with the approximations to mathematical curvature, (2) an easily measured quantity, (3) robust to noisy boundaries, (4) useful to detect features in varying sizes.

It is an iterative approach where at each iteration a two phased analysis is performed. Starting the analysis at a coarser level of resolution, the method increases the analysis resolution at each iteration step considering the field distribution within the determined candidate regions. The reference for evaluating the field mean values within the regions is the distributions obtained from an infinite-length straight line. By means of the reference shape, features in different size in the same image can easily be identified automatically, requiring no user based intervention. The proposed method is utilized in determining the registration points which are used in over-complete graph representations of line drawings used in the solution of part embedding problem discussed in Chapter 2. In addition to that, the preliminary experiments on some contour shapes show that the method has a high potential to be useful in pattern classification and shape matching.

CHAPTER 5

CONCLUSION

The motivation behind the research described in this thesis was finding a generic computational solution for part relations of shapes in coherence with the philosophy of the shape grammar formalism, mostly missing in the existing part embedding solutions. Previous solutions to the part embedding problem either assumed that the shapes are composed of rectilinear spatial elements such as points, straight lines, or that they can be represented in a specific parametric form such as cubic or quadratic Bezier curves. Although the existing solutions are inspiring for the assumed parametric forms, we believe that making such presumptions about spatial primitives of shapes is as restrictive as defining shapes with some predefined set of parts. Both presumptions similarly harm the part embedding idea in rule applications in the shape grammar formalism. Instead, the research developed in the context of this study is concerned about exploring a generic computational solution for the part embedding problem which will be free from either predefined analytic abstractions related with shape primitives or predefined decompositions of shapes.

Two generic solutions are developed considering these issues. In the first solution, which is provided in Chapter 2, a novel graph data structure is explored to temporarily represent the boundary elements of shapes with the relations among them. Equipped with the associated algorithms, this data structure enables a systematic search for parts. The part detection is based on the over-complete character of the shape graph representation as well as the user-defined/external constraints. The decomposition of an initial shape is not fixed but alternates depending on the applied rules and externally provided constraints. Boundary elements of the perceived wholes are the key elements for representing shapes and defining the relations among the temporary parts. These points belong to a lower level algebra which serve as the registration points in part searches. In this way, shape is considered as it is, while - when necessary - only the mapping function to the lower level algebra is modified.

The over-complete graph is constructed by operating on multiple but temporary representations in the label algebra. It facilitates the passage between the image representations and shapes by means of its attributes. The structure of an over-complete graph is also appropriate for defining a number of different perceptual wholes together, in addition to the maximal elements.

The approach presented in the first solution utilizes shape algebras and part relations true to visual thinking. It does not rely on any geometric notion or symbolic representation. It works with perceptual wholes and without fixed primary features unless specified by the user. In the implementation of the method, as a default case, the critical points are utilized to define boundaries of parts; however, they may always be changed according to the user/designer intentions.

While searching for the first solution to the part embedding problem, a novel approach is developed to identify the critical points for the line drawings. Such points have helped defining a generic mechanism to map a given shape to a lower level algebra to determine the set of registration points for the part searches. The novelty of this method is in its utilization of the continuous field for evaluating criticality of the points instead of the widely used measures of criticality such as extrema of curvature. The method can detect the critical features in different sizes in an automatic manner with a two phased iterative analysis. This method proved itself useful in the part embeddings when these points are used as the default set of part boundaries in representing the perceptual wholes. Moreover, a set of initial experiments show the potential of this method in representing shape contours for the more general shape matching and pattern classification applications.

Consistent with the principles of the first solution, an alternative solution is explored during the course of this study, which supports the creative design processes. This solution is discussed in Chapter 3. The second method explores the search space which is induced by an initial shape for embedding a part into it with an evolutionary approach. Staying true to the continuous nature of shapes, representation is carried out in a higher level algebra this time, with weighted shapes in the planar domain. This representation provides many advantages: (1) the continuous character of the weights help defining part embedding operations true to the continuous character of shapes more naturally, (2) any distinction between line shapes and curve shapes disappeared in this representation, (3) instead of the external or abstract constraints, the weight values are utilized as registration marks.

Being an alternative solution to the first one, the part embedding problem is approached in the second solution with an evolutionary method via an implementation of the genetic algorithm. The approach is unique in that a part searched within an initial shape is considered as a whole, rather than a composition of parts. This representation enabled working with any categories of shapes such as lines, curves, etc., making the distinction among them irrelevant. Various runs of the implementation on line drawings such as Seljuk patterns show that parts are perceived without predefinitions. In addition to this, wholes that are perceived by the eye despite some missing parts can also be detected as parts and hence, the method introduces emergence based on designer defined constraints.

It is widely accepted that human vision processes the visual data by parts and many previous conjectures in cognitive vision support part based representations of objects ([8], [22], [42], [43], [9]). In this respect, different approaches impose different regularities on determining parts: some impose regularity on the shapes of parts ([7]), while most others, in parallel with the idea stated in the seminal work of Attneave ([4]), impose regularity on the part boundaries ([24]). Different from the previous approaches, the proposed methods in this research bring out two insights to the representation of shapes: (1) a pragmatic way of defining boundaries of parts, which depends on user defined constraints and is temporary, (2) considering the shape as a whole, where there are no predefined part boundaries. The representation of shapes with perceptual wholes is common in both of the approaches which makes them coherent with the philosophy of the shape grammar formalism. Intriguingly, recent evidence from neuroscience ([49]) and psychophysics ([16]) support our approaches by their implication that parts are not defined with strict boundaries but are represented in a continuous manner.

5.1 Future Works

As a future work, a general purpose shape grammar interpreter can be developed utilizing the solutions provided for part embedding. Considering the need for a general solution for shape grammar interpreters, the solutions provided for part embedding could reinitiate the research in the field.

Specifically, the evolutionary approach introduced as the second solution is very suitable for the analysis of the complex line drawings, like Seljuk tiles. We believe that development of an analysis system supporting the designer's choice of constraints using this solution would support creative thinking and hence help the creative design process a lot.

Another research direction would be utilization of the critical region analysis method in a shape matching framework and improve the method further according to the needs that are specific to that problem domain. Initial experiments with the real object boundaries show that the method has a high potential to be useful in shape matching and pattern recognition applications.

An alternative research direction is the parallelization of the critical point detection method. Since the analysis is performed for each pixel/point of an image, the execution time is proportional to the number of pixels within an image and hence costly for complex line drawings. The nature of the developed solution is very suitable to parallelization, because each analysis window works in a local neighborhood independently. Considering the power of parallel processors available within the commodity graphics processing units today, the parallel solution to the critical point detection would be more time efficient and be available for ordinary desktop computers.

REFERENCES

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

[2] Agarwal M., Cagan J., Stiny G. A micro language: generating MEMS resonators by using a coupled form - function shape grammar. *Environment and Planning B: Planning and Design*, **27(4)**:615–626, 2000.

[3] Ansari N., Huang K.W. Non-parametric dominant point detection. *Pattern Recognition*, **24**:849-862, 1991.

[4] Attneave F. Some Informational Aspects of Visual Perception. *Psycological Review*, **61(3)**:183-193, 1954.

[5] Bakirer O. Selcuklu Oncesi ve Selcuklu Donemi Anadolu Mimarisinde Tugla Kullanimi. *ODTU Mimarlik Fakultesi Basim Isligi*, Ankara, Turkey, 1981.

[6] Beaua V. and Singera M. Reduced resolution and scale space for dominant feature detection in contours. *Pattern Recognition*, **34(2)**:287–297, 2001.

[7] Biederman I. Recognition-by-components: A theory of human image understanding. *Psychological Review*, **94**:115–117, 1987.

[8] Binford T.O. Visual perception by computer. In *IEEE Systems Science and Cybernetics Conference*, Miami, FL, 1971.

[9] Brooks R.A. Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence*, **17**:205–244, 1981.

[10] Brown K. N., McMahon C. A., Sims-Williams J. H. A formal language for the design of manufacturable objects. In *Formal Design Methods for CAD (B-18)*, pages 135–155, Amsterdam, 1994.

[11] Cagan J. Engineering shape grammars. In *Formal Engineering Design Synthesis*, Cambridge University Press, pages 65–92, Eds. E. K. Antonsson, J. Cagan, 2001.

[12] Chase S.C. Shape and shape grammars: from mathematical model to computer implementation. *Environment and Planning B: Planning and Design*, **16**:215–242, 1989.

[13] Chau C.P. and Siu W.S. New nonparametric dominant point detection algorithm. *IEE Proc. Vision Image Signal Process.*, **148(5)**:363–374, 2001.

[14] Chau H.H., Chen X., McKay A. and Pennington A. Evaluation of a 3D shape grammar implementation. In *First International Conference on Design Computing and Cognition*, Cambridge, MA, 2004.

[15] Chui H. and Rangarajan A. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding In Nonrigid Image Registration*, **89(2-3)**:114–141, 2003.

[16] Cohen E. and Singh M. Geometric determinants of shape segmentation: Tests using segment identification. *Vision Research*, **47(22)**:2825–2840, 2007.

[17] Fischler M.A. and Bolles R.C. Perceptual Organization and Curve Partitioning. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **8(1)**:100–105, 1986.

[18] Fischler M.A. and Wolf H.C. Locating Perceptually Salient Points on Planar Curves. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **16(2)**:113–129, 1994.

[19] Flemming U. More than the sum of parts: the grammar of Queen Anne houses. *Environment and Planning B: Planning and Design*, **14(3)**:323–350, 1987.

[20] Gips J. A syntax-directed program that performs a three-dimensional perceptual task. *Pattern Recognition*, **6**:189–199, 1974.

[21] Gips J. Shape Grammars and their Uses. *Birkhauser-Basel*, 1975.

[22] Guzman A. Analysis of curved line drawings using context and global information. *Machine Intelligence 6*, pages: 325–375, Edinburg: Edinburg University Press, 1971.

[23] Heisserman J. Generative Geometric Design. *IEEE Computer Graphics and Applications*, **14(2)**:37–45, 1994.

[24] Hoffman D., and Richards W. Parts of Recognition. *Cognition*, **18**:65–96, 1984.

[25] Hoffman D. Representing Shapes for Visual Recognition. *Ph.D. Dissertation, MIT*, 1983.

[26] Jowers I. Computation with Curved Shapes: Towards Freeform Shape Generation in Design. *Ph.D. Thesis in the Department of Design and Innovation, The Open University*, United Kingdom, 2006.

[27] Jowers I. and Earl C. The construction of curved shapes. *Environment and Planning B: Planning and Design*, **37**:42–58, 2010.

[28] Keles H., Ozkar M., Tari S. Embedding Shapes Without Predefined Parts. *Environment and Planning B: Planning and Design*, **(in press)**, 2010.

[29] Knight T.W. The generation of Hepplewhite-style chair-back designs. *Environment and Planning B: Planning and Design*, **7**:227–238, 1980.

[30] Knight T.W. Transformations of De Stijl art: the paintings of Georges Vantongerloo and Fritz Glarner. *Environment and Planning B: Planning and Design*, **16**:51–98, 1989.

[31] Knight T. W. Transformations in Design: A formal approach to stylistic change and innovation in the visual arts. *Cambridge University Press*, Cambridge, UK, 1994.

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

[33] Koning H. and Eizenberg J. The language of the prairie: Frank Lloyd Wright's Prairie Houses. *Environment and Planning B: Planning and Design*, **8**:295–323, 1981.

[34] Krishnamurti R. The arithmetic of shapes. *Environment and Planning B: Planning and Design*, **7**:463–484, 1980.

[35] Krishnamurti R. The construction of shapes. *Environment and Planning B: Planning and Design*, **8**:5–40, 1981.

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

[37] Krishnamurti R. The maximal representation of a shape. *Environment and Planning B: Planning and Design*, **19**:267–288, 1992.

[38] Krishnamurti R. The arithmetic of a maximal planes. *Environment and Planning B: Planning and Design*, **19**:431–464, 1992.

[39] Krishnamurti R. and Stouffs R. The boundary of a shape and its classification. *Journal of Design Research*, **4(1)**:28, 2004.

[40] Latecki L. and Lakamper R. Convexity Rule for Shape Decomposition Based on Discrete Contour Evolution. *Computer Vision and Image Understanding*, **73(3)**:441–454, 1999.

[41] Marji M. and Siy P. Polygonal representation of digital planar curves through dominant point detection - a nonparametric algorithm. *Pattern Recognition*, **37**:2113–2130, 2004.

[42] Marr D. Analysis of occluding contour. *Proceedings of the Royal Society of London, Series B, 197*, pages: 441–475, 1977.

[43] Marr D. and Nishihara, H.K. Representation and recognition of three dimensional shapes. *Proceedings of the Royal Society of London, Series B, 200*, pages: 269–294, 1978.

[44] McCormack J.P. and Cagan J. Supporting designer's hierarchies through parametric shape recognition. *Environment and Planning B: Planning and Design*, **29**:913–931, 2002.

[45] McCormack J.P. and Cagan J. Curve-based shape matching: supporting designer's hierarchies through parametric shape recognition of arbitrary geometry. *Environment and Planning B: Planning and Design*, **33**:523–540, 2006.

[46] McGill M.C. A visual approach for exploring computational design. *M.Sc. in Architecture Studies thesis in School of Architecture and Planning of Massachusetts Institute of Technology*, Cambridge, MA, 2001.

[47] Moktarian F. and Macworth A.K. A theory of multiscale-based shape representation for planar curves. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **14**:789–805, 1992.

[48] Ozkar M. and Lefford N. Modal Relationships as Stylistic Features: Examples from Seljuk and Celtic Patterns. *Journal of American Society of Information Science and Technology (JASIST)*, **57**:1551-1560, 2006.

[49] Pasupathy A., Connor C. E. Shape representation in area V4: Position-specific tuning for boundary conformation. *J. Neurophysiol.* **86**:2505–2519, 2001.

[50] Pei S-C. and Lin C-N. The detection of dominant points on digital curves by scale-space filtering. *Pattern Recognition*, **25**:1307–1314, 1992.

[51] Piazzalunga U. and Fitzhorn P. Note on a three-dimensional shape grammar interpreter. *Environment and Planning B: Planning and Design*, **25(1)**:11–30, 1998.

[52] Prats M., Earl C., Garner S., Jowers I. Shape Exploration of Designs in a Style: Toward Generation of Product Designs. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **20**:201–215, 2006.

[53] Pugliese M.J. and Cagan J. 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, 2002.

[54] Ray B.K. and Ray K.S. Detection of significant points and polygonal approximation of digitized curves. *Pattern Recognition Letters*, **12**:443–452, 1992.

[55] Ray B.K. and Pandyan R. ACORD-an adaptive corner detector for planar curves. *Pattern Recognition*, **36**:703–708, 2003.

[56] Rattatangsi A. and Chin R.T. Scale based detection of corners of planar curves. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **14**:430–449, 1992.

[57] Rosin P.L. Representing curves at their natural scales. *Pattern Recognition*, **25**:1315–1325, 1992.

[58] Shea K. and Cagan J. The design of novel roof trusses with shape annealing: assessing the ability of a computational method in aiding structural designers with varying design intent. *Design Studies*, **20**:3–23, 1999.

[59] Stiny G. and Gips J. Shape Grammars and the Generative Specification of Painting and Sculpture. In *Information Processing '71*, pages 1460–1465, 1972.

[60] Stiny G. Ice Ray: A Note on the Generation of Chinese Lattice Design. *Environment and Planning B: Planning and Design*, **4**:89–98, 1977.

[61] Stiny G. and Mitchell W.J. The Palladian Grammar. *Environment and Planning B: Planning and Design*, **5**:5–18, 1978.

[62] Stiny G. Spatial relations and grammars. *Environment and Planning B: Planning and Design*, **9(1)**:113–114, 1982.

[63] Stiny G. The algebras of design. *Research in Engineering Design*, **2**:171–181, 1991.

[64] Stiny G. Weights. *Environment and Planning B: Planning and Design*, **19**:413–430, 1992.

[65] Stiny G. Boolean-algebras for shapes and individuals. *Environment and Planning B: Planning and Design*, **20(3)**:359–362, 1993.

[66] Stiny G. Shape Rules: Closure, Continuity, and Emergence. *Environment and Planning B: Planning and Design*, **21**:49–78, 1994.

[67] Stiny G. Pictorial and Formal Aspects of Shape and Shape Grammars. *Birkhauser-Basel*, 1975.

[68] Stiny G. How to calculate with shapes. In *Formal Engineering Design Synthesis*, Cambridge University Press, pages 20–64, Eds. E. K. Antonsson, J. Cagan, 2001.

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

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

[71] Tari S. Shah J. and Pien H. Extraction of shape skeletons from grayscale images. *Comput. Vis. Image Underst.*, **66(2)**:133–146, 1997.

[72] Tari S. Extracting parts of 2D shapes using local and global interactions simultaneously. *Handbook of Pattern Recognition and Computer Vision*, World Scientific, Edt. C. Chen, 2010.

[73] Teh C. and Chin R. On the detection of dominant points on digital curve. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **11(8)**:859–872, 1989.

[74] Urdiales C., Bandera A., Sandoval F. Nonparametric planar shape representation based on adaptive curvature functions. *Pattern Recognition*, **35**:43–53, 2002.

[75] Wang Y.F. and Duarte J.P. Automatic generation and fabrication of designs. *Automation in Construction*, **11**:291–302, 2002.

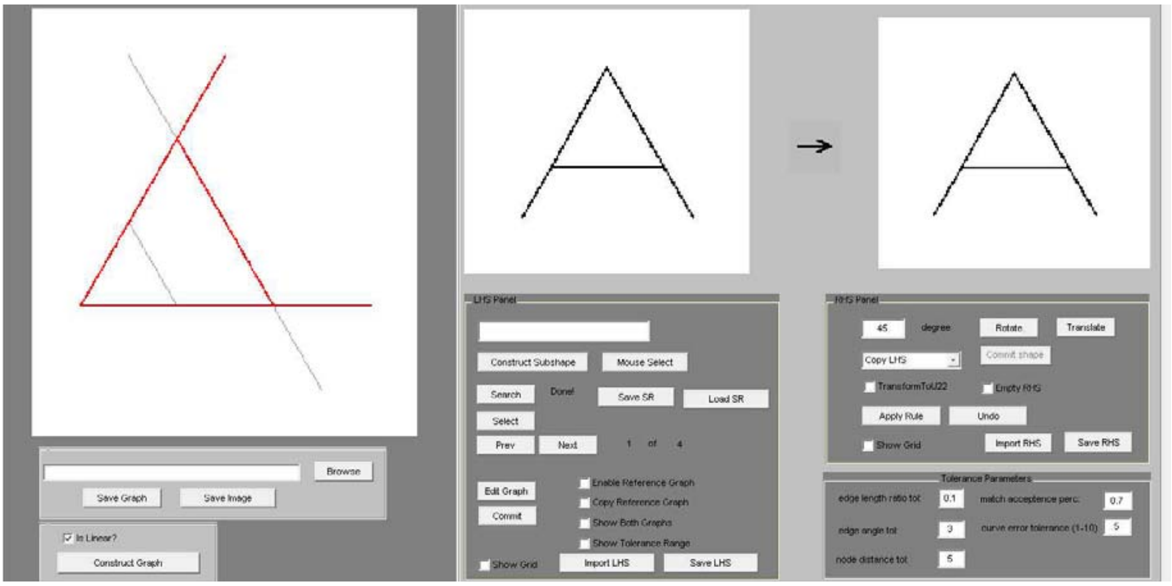
[76] Wells A. B. Grammars for engineering design. *Ph.D. Dissertation, California Institute of Technology*, Pasadena, CA, 1994.

[77] Wu W-Y. An adaptive method for detecting dominant points. *Pattern Recognition*, **36**:2231–2237, 2003.

APPENDIX A

Sample Shape Embeddings

A.1 Linear Sub-shape Embedding GUI Samples



☑ Is Linear?

Done!

1 of 4

☐ Enable Reference Graph
☐ Copy Reference Graph
☐ Show Both Graphs
☐ Show Tolerance Range
☐ Show Grid

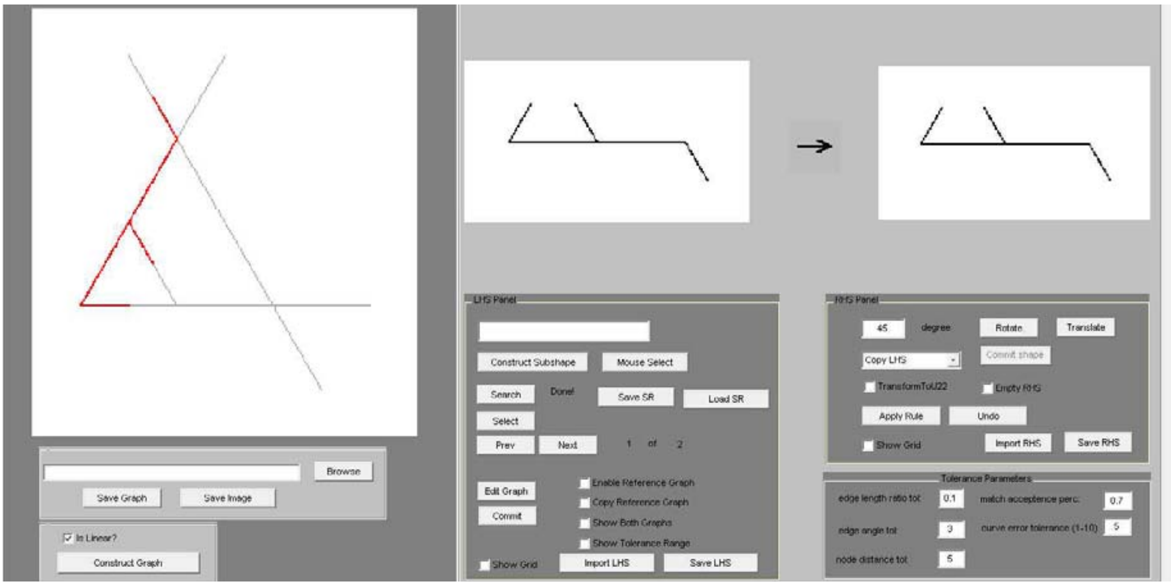
45 degree

Copy LHS

☐ TransformToU22
☐ Empty RHS
☐ Show Grid

edge length ratio tol 0.1
match acceptence perc. 0.7
edge angle tol 3
curve error tolerance (1-10) 5
node distance tol 5

Figure A.1: One of the four embedding results for the examplary case in Figure 2.17-(a).



☑ Is Linear?

Done!

1 of 2

☐ Enable Reference Graph
☐ Copy Reference Graph
☐ Show Both Graphs
☐ Show Tolerance Range
☐ Show Grid

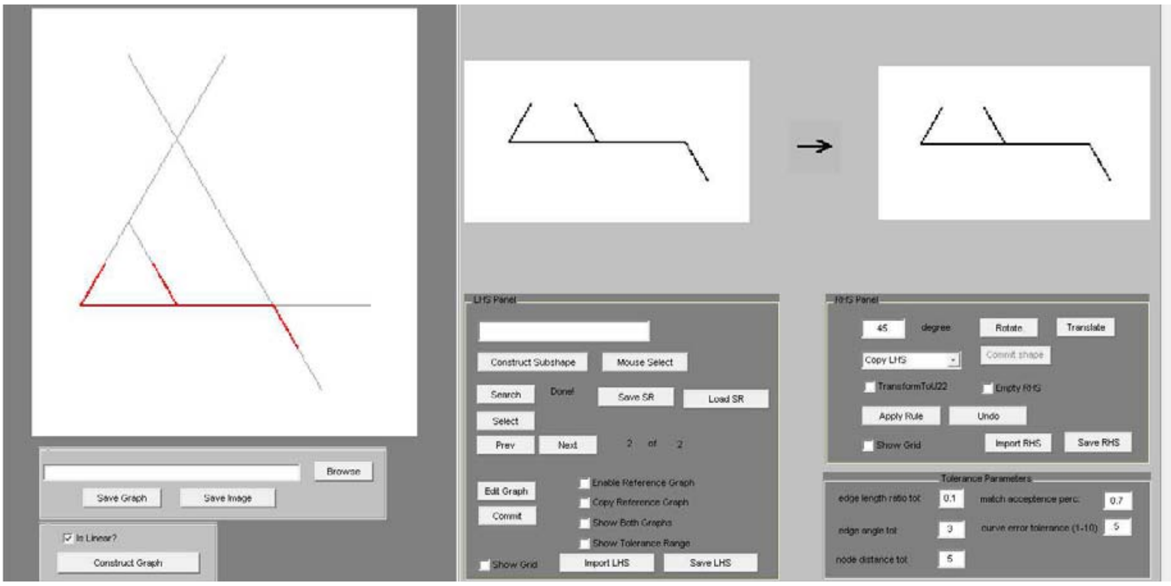
45 degree

Copy LHS

☐ TransformToU22
☐ Empty RHS
☐ Show Grid

edge length ratio tol 0.1
match acceptence perc. 0.7
edge angle tol 3
curve error tolerance (1-10) 5
node distance tol 5

Figure A.2: One of the two embedding results for the examplary case in Figure 2.17-(b).



☑ Is Linear?

Done!

2 of 2

☐ Enable Reference Graph
☐ Copy Reference Graph
☐ Show Both Graphs
☐ Show Tolerance Range
☐ Show Grid

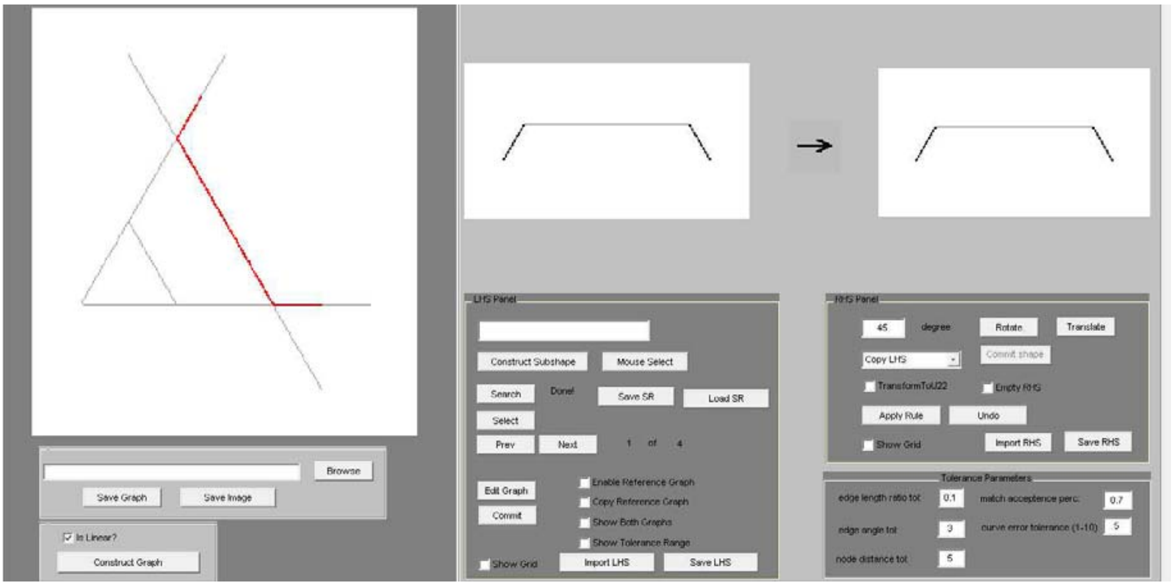
45 degree

Copy LHS

☐ TransformToU22
☐ Empty RHS
☐ Show Grid

edge length ratio tol 0.1
match acceptence perc. 0.7
edge angle tol 3
curve error tolerance (1-10) 5
node distance tol 5

Figure A.3: One of the two embedding results for the examplary case in Figure 2.17-(b).



☑ Is Linear?

Done!

1 of 4

☐ Enable Reference Graph
☐ Copy Reference Graph
☐ Show Both Graphs
☐ Show Tolerance Range
☐ Show Grid

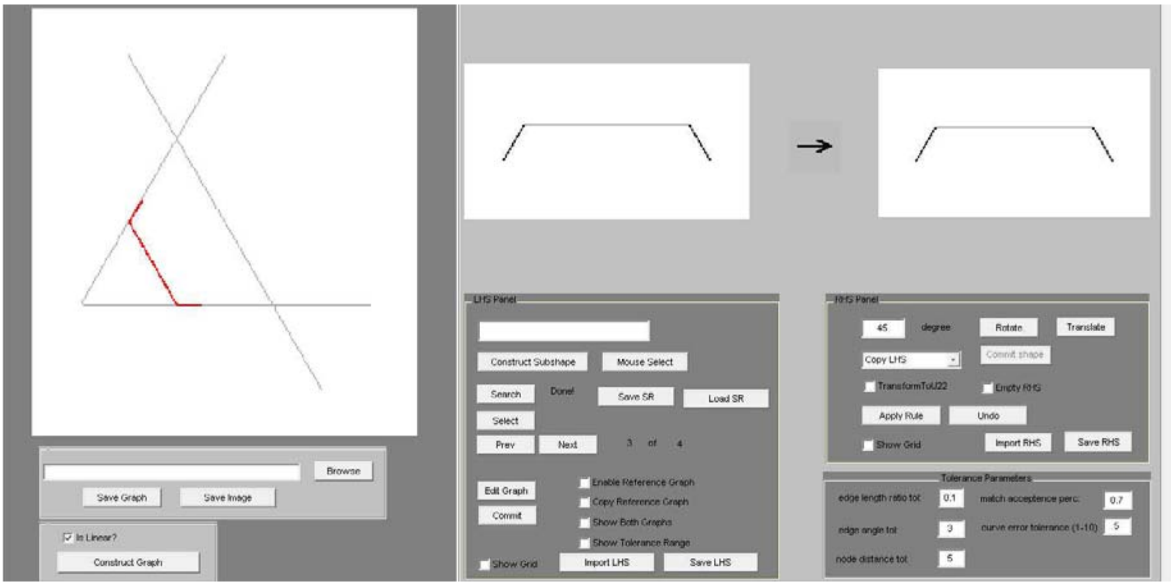
45 degree

Copy LHS

☐ TransformToU22
☐ Empty RHS
☐ Show Grid

edge length ratio tol: 0.1
edge angle tol: 3
node distance tol: 5
match acceptence perc: 0.7
curve error tolerance (1-10): 5

Figure A.4: One of the four embedding results for the examplary case in Figure 2.17-(c).



Done!

3 of 4

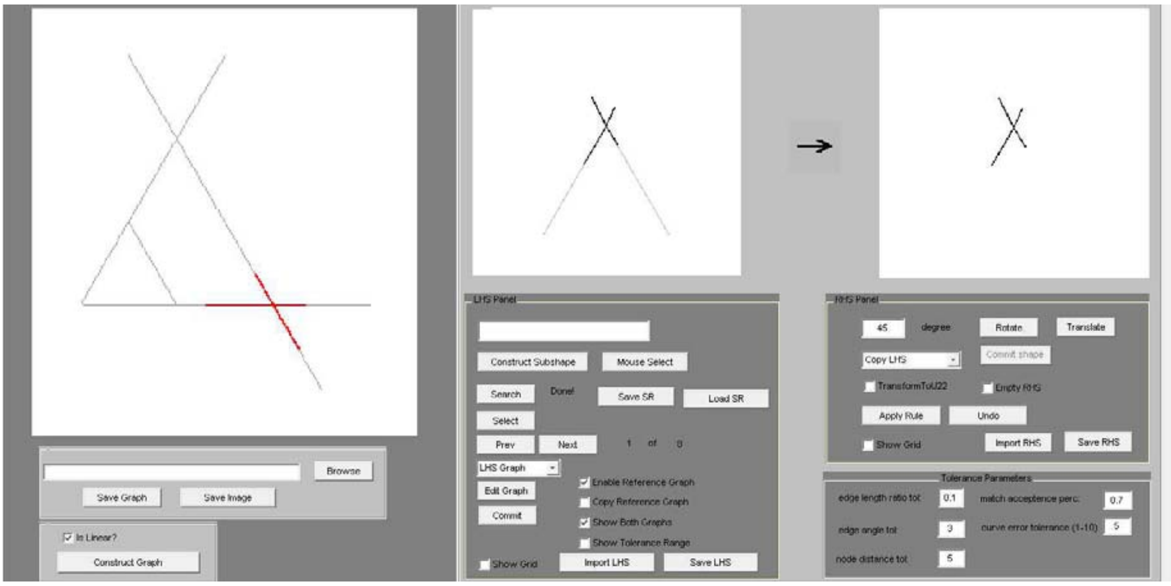
☐ Enable Reference Graph
☐ Copy Reference Graph
☐ Show Both Graphs
☐ Show Tolerance Range
☐ Show Grid

45 degree
Copy LHS
☐ TransformToU22
☐ Empty RHS
☐ Show Grid

☑ Is Linear?

edge length ratio tol: 0.1
match acceptence perc: 0.7
edge angle tol: 3
curve error tolerance (1-10): 5
node distance tol: 5

Figure A.5: One of the four embedding results for the examplary case in Figure 2.17-(c).



Done!

1 of 8

LHS Graph

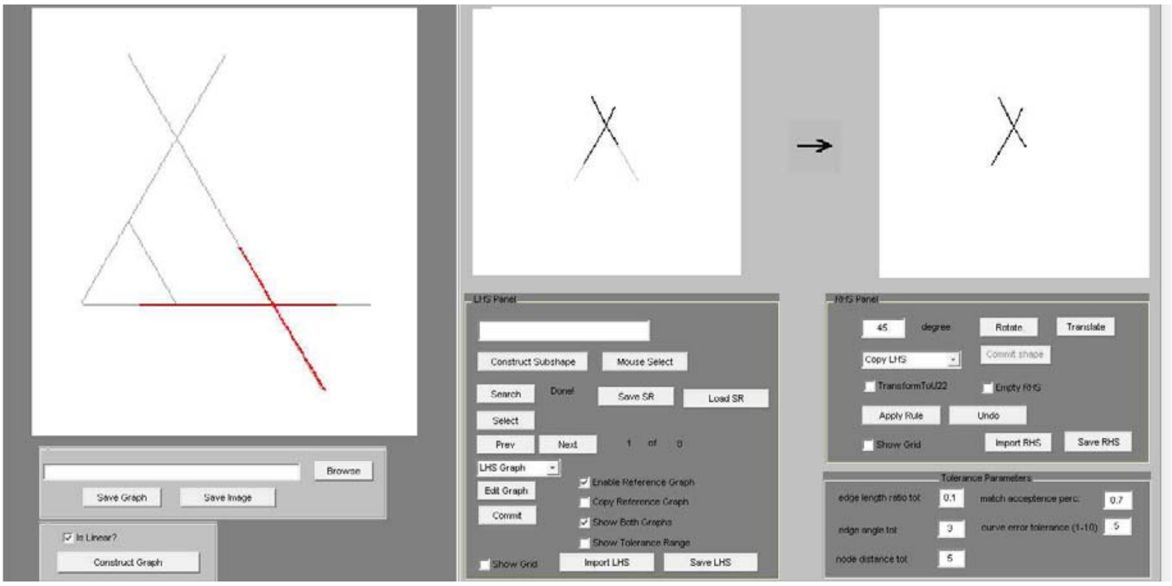
☑ Enable Reference Graph
☐ Copy Reference Graph
☑ Show Both Graphs
☐ Show Tolerance Range
☐ Show Grid

45 degree
Copy LHS
☐ TransformToU22
☐ Empty RHS
☐ Show Grid

☑ Is Linear?

edge length ratio tol: 0.1
match acceptence perc: 0.7
edge angle tol: 3
curve error tolerance (1-10): 5
node distance tol: 5

Figure A.6: One of the eight embedding results for the examplary case in Figure 2.17-(d), with the user defined constraint.



☑ Is Linear?

Done!

1 of 8

☑ Enable Reference Graph
☐ Copy Reference Graph
☑ Show Both Graphs
☐ Show Tolerance Range
☐ Show Grid

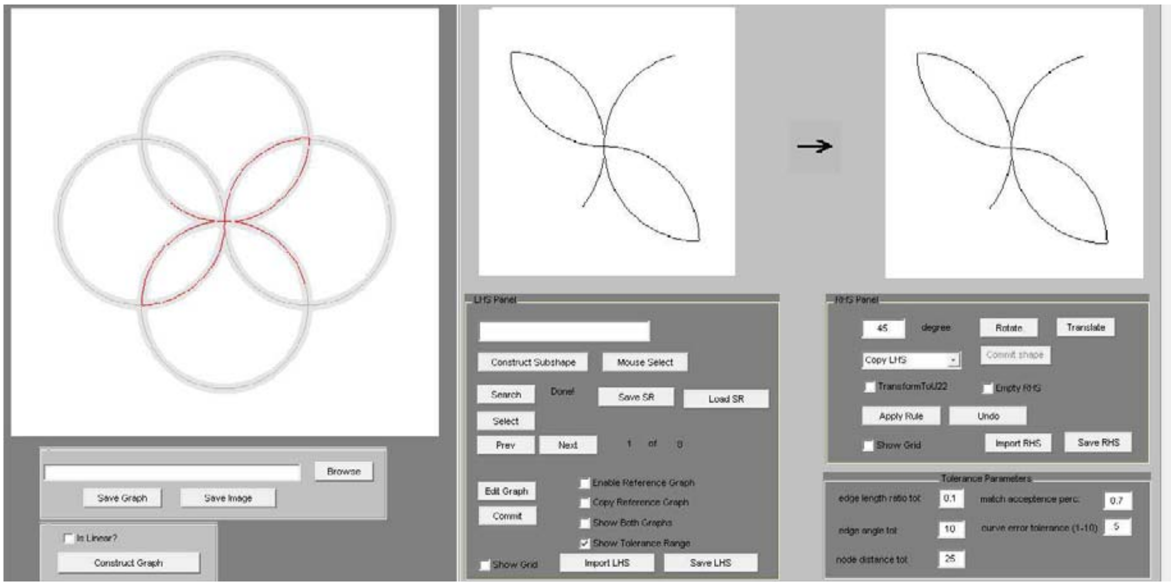
45 degree

☐ TransformToU22
☐ Empty RHS
☐ Show Grid

edge length ratio tol: 0.1
match acceptence perc: 0.7
edge angle tol: 3
curve error tolerance (1-10): 5
node distance tol: 5

Figure A.7: One of the eight embedding results for the examplary case in Figure 2.17-(d), with a different user defined constraint.

A.2 Generalized Sub-shape Embedding GUI Samples



Done!

1 of 8

45 degree

Copy LHS

☐ TransformToU22

☐ Empty RHS

☐ Show Grid

☐ Enable Reference Graph

☐ Copy Reference Graph

☐ Show Both Graphs

☑ Show Tolerance Range

☐ Show Grid

☐ Is Linear?

edge length ratio tol: 0.1

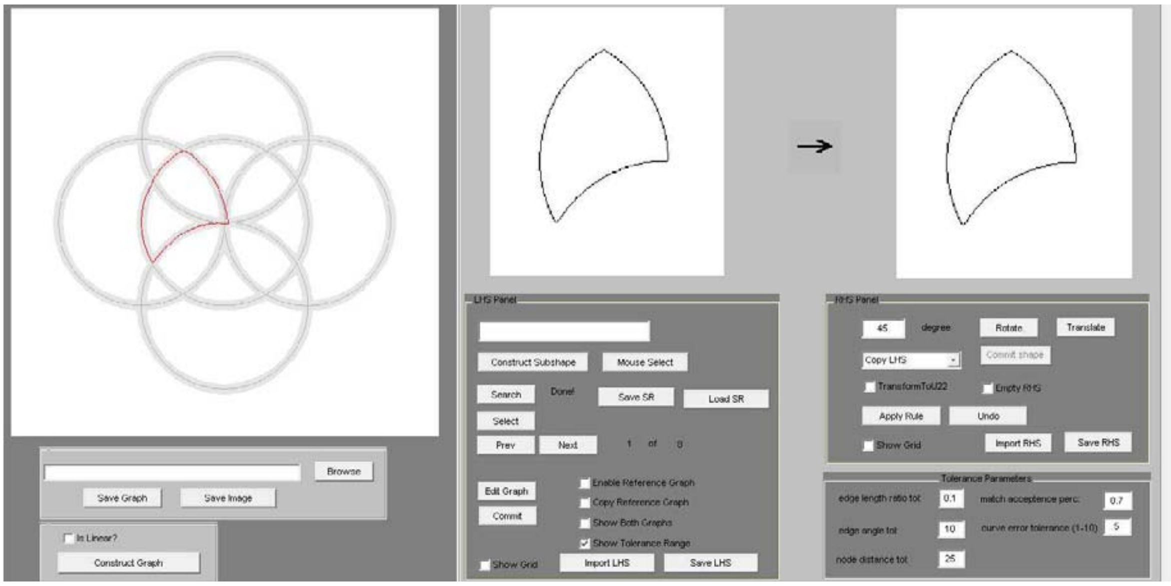
match acceptence perc: 0.7

edge angle tol: 10

curve error tolerance (1-10): 5

node distance tol: 25

Figure A.8: One of the eight embedding results for a sample shape.



Done!

1 of 8

45 degree

Copy LHS

☐ TransformToU22

☐ Empty RHS

☐ Show Grid

☐ Enable Reference Graph

☐ Copy Reference Graph

☐ Show Both Graphs

☑ Show Tolerance Range

☐ Show Grid

☐ Is Linear?

edge length ratio tol: 0.1

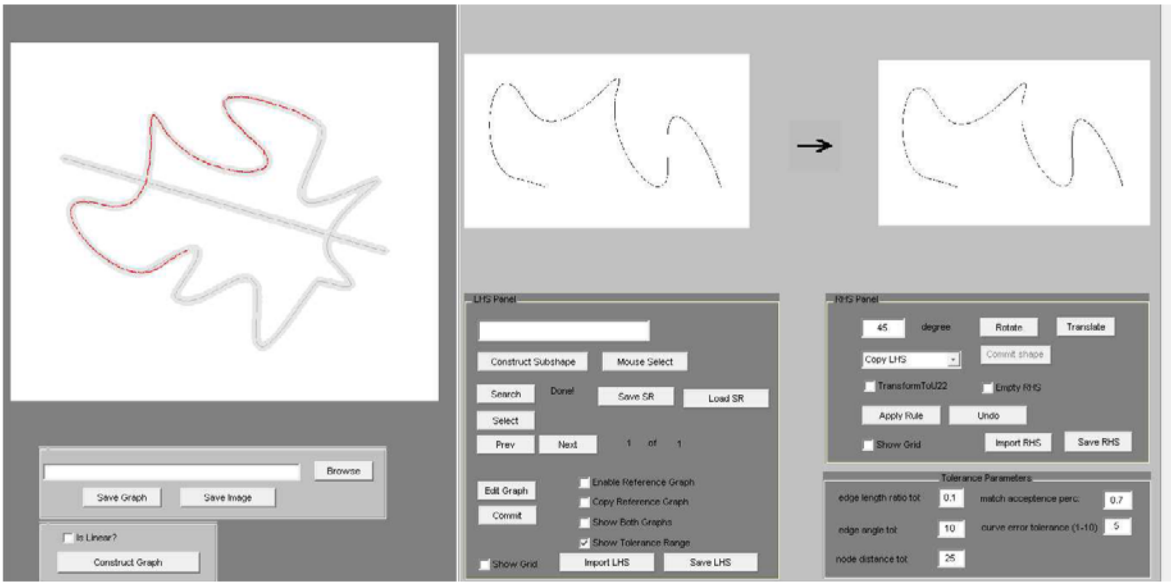
match acceptence perc: 0.7

edge angle tol: 10

curve error tolerance (1-10): 5

node distance tol: 25

Figure A.9: One of the eight embedding results for a sample shape.



☐ Is Linear?

Done!

1 of 1

☐ Enable Reference Graph
☐ Copy Reference Graph
☐ Show Both Graphs
☑ Show Tolerance Range
☐ Show Grid

45 degree

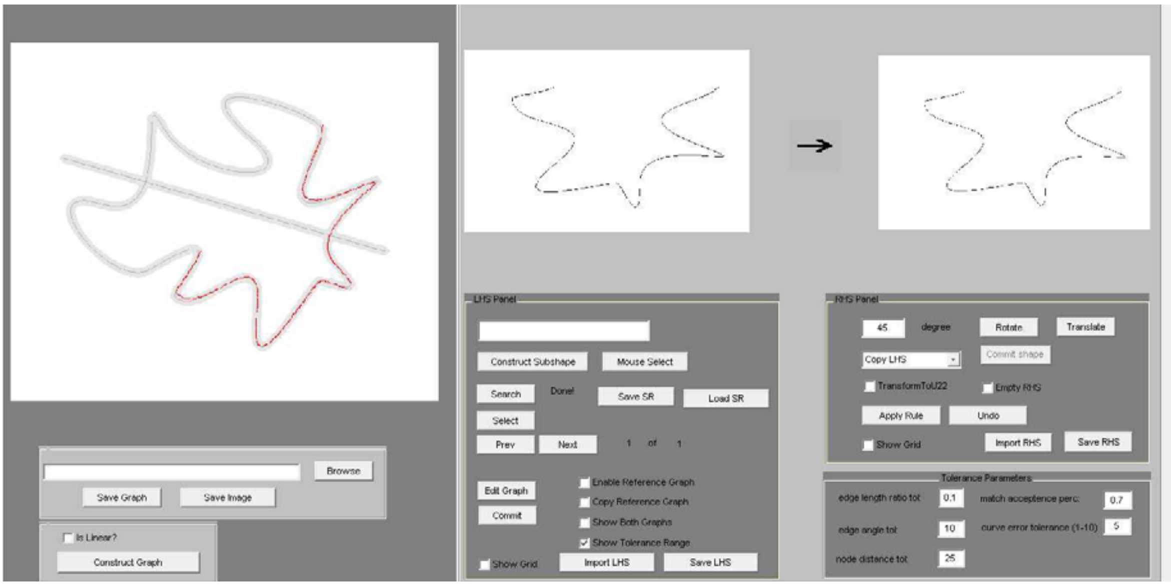
Copy LHS

☐ TransformToU22
☐ Empty RHS
☐ Show Grid

Tolerance Parameters

edge length ratio tol	0.1	match acceptence perc:	0.7
edge angle tol	10	curve error tolerance (1-10)	5
node distance tol	25		

Figure A.10: One embedding results for a sample shape.



☐ Is Linear?

Done!

1 of 1

☐ Enable Reference Graph
☐ Copy Reference Graph
☐ Show Both Graphs
☑ Show Tolerance Range
☐ Show Grid

45 degree

Copy LHS

☐ TransformToU22
☐ Empty RHS
☐ Show Grid

edge length ratio tol 0.1
match acceptence perc: 0.7
edge angle tol 10
curve error tolerance (1-10) 5
node distance tol 25

Figure A.11: One embedding results for a sample shape.

APPENDIX B

Implementation of the Edge Strength Function

The edge strength function, w, given in Chapter 3-equation 3.1 is computed by the following equation:

$$\frac{\partial w}{\partial t} = (\nabla^2 - \lambda)w,$$

where,

$$\lambda = \frac{1}{\rho^2}$$

In the discrete case this equation is solved using the finite difference approximations. For the space derivative, central differences are used. Forward differencing is used for the time derivative. As a result of the discretization of this equation, we obtain the following equation:

$$\frac{w_{i,j}^{k+1} - w_{i,j}^{k}}{\Delta t} = w_{i+1,j}^{k} + w_{i-1,j}^{k} + w_{i,j+1}^{k} + w_{i,j-1}^{k} - 4w_{i,j}^{k} - \lambda w_{i,j}^{k}$$

where, i and j are the space indices and k is the iteration number.

The diffusion process starts with the given initial image, which is used as w^0 in the equation above. In the image, the pixels belonging to the shape S is set to 1 and kept constant during the iterations. At the image boundaries, the function values are assumed to equal to the nearest image border value. Δt should be less than 0.25 for this equation to converge. In our experiments we set Δt to 0.2.

VITA

PERSONAL INFORMATION

NAME, SURNAME:	Hacer Yalım Keleş
DATE OF BIRTH:	April 11, 1979
PLACE OF BIRTH:	Ankara, TURKEY
MARITAL STATUS:	Married, (have a daughter)
PHONE:	+905054587780
E-MAIL:	hacerkeles@gmail.com
WEB ADRESS:	www.haceryalimkeles.com

DEGREES

M.Sc.	2005	Computer Engineering, METU
B.Sc.	2002	Computer Engineering, METU

INDUSTRY R&D EXPERIENCE

MITT: Ministry of Industry and Trade of Turkey.

TUBITAK-UZAY: Space Technologies Research Institute of the Scientific and Technological Research Council of Turkey.

Year	Place	Project Name	Enrollment
April 2010 - April 2011	Funded by MITT	OrtakTEST Project	Project Manager
2006-2007	TUBITAK-UZAY	RTUK-SKAAS	Senior Researcher
2004-2006	TUBITAK-UZAY	TIBBİ DİKTE	Senior Researcher
2002-2004	TUBITAK-UZAY	MPEG-7 LLAD	Researcher
2000-2002	TUBITAK-UZAY	Various Small Scale Projects	Part-time Researcher

PUBLICATIONS

Journals

1. Hacer Yalim Keles, Mine Ozkar, Sibel Tari. Weighted Shapes for Embedding Perceived Wholes, *Environment and Planning B: Planning and Design*. (submitted, May 2010).
2. Hacer Yalim Keles, Mine Ozkar, Sibel Tari. Embedding Shapes without Predefined Parts, *Environment and Planning B: Planning and Design*, in press.
3. Hacer Yalim Keles, Alphan Es, Veysi Isler. Acceleration of Direct Volume Rendering with Programmable Graphics Hardware, *The Visual Computer: International Journal of Computer Graphics*, **23(1)**:15-24, November, 2006.

International Conferences

1. Hacer Yalim Keles, Mine Ozkar, Sibel Tari. Revisiting Shape Embedding, *27'th Education and Research in Computer Aided Architectural Design in Europe (eCAADe) Conference*, 16-19, September 2009, İstanbul, Turkey.
2. Gokce Yildirim, Hacer Yalim Keles, Veysi Isler. Three Dimensional Smoke Simulation on Programmable Graphics Hardware, *International Conference on Parallel Computational Fluid Dynamics*, 21-24, May 2007, Turkey.
3. Alphan Es, Hacer Yalim Keles, Veysi Isler. Accelerated Volume Rendering with Homogeneous Region Encoding using EACD on GPU, *6'th Eurographics Symposium on Parallel Graphics and Visualization*, 11-12, May 2006, Braga.