

A QUALITATIVE
APPROACH FOR
REPRESENTATION AND
MATCHING OF SHAPES

By

Muhammad Ahmed

A thesis submitted in partial fulfillment of
the requirements for the degree of

Masters of Science in Digital Media

University of Bremen

2007

Supervisor

Dr. rer. nat. Thomas Barkowsky

Second Supervisor

Prof. Dr. Kerstin Schill

September 6, 2007



UNIVERSITY OF BREMEN

ABSTRACT

A Qualitative Approach for Representation and Matching of Shapes

By Muhammad Ahmed

Supervisor:

Dr. Thomas Barkowsky
Department of Informatics

Shape representation is an important interest of qualitative spatial reasoning for studying objects behavior and their interaction with other objects. Although there have been several approaches in this area, these approaches are limited in its precision to describe shapes at different granularity levels. In each approach, we find examples of shapes that are cognitively distinguishable but unable to be distinguished through these approaches. In this thesis, we present an outline-based approach of qualitative shape representation that would take into consideration a more detail positional and qualitative size information to successfully describe such shapes. Our work will be an extension of an existing approach— reference points based representation [Museros, L. and Escig, M. T. 2003]. The extended approach implements particular features, including partial matching and fine versus coarse matching, allowing it to describe broad range of shapes. However, our main focus in this thesis is geographic shapes for GIS applications such as shapes that represent boundaries of countries, rivers and lakes in a map.

TABLE OF CONTENTS

1.Introduction	1
1.2 Thesis Statement.....	3
2.Qualitative Shape Representation, State of the art	4
2.1 Cohn’s Representation.....	5
2.2 Contour Codons	7
2.3 Extremum Primitive.....	9
2.4 Qualitative Outline Theory	10
2.5 Tripartite Line Tracks Representation	13
2.6 Reference Point Based Representation	15
2.7 Discussion	20
2.8. Limitations of Museros and Escrig’s Scheme	22
3. Extended Reference Point Based Representation	26
3.1 Extensions.....	26
3.1.1 Description of Angles.....	27
3.1.2 Description of Lengths.....	30
3.1.3 Description of Curvature type for Curvilinear Segments	32
3.2 Complete Description of Shapes	33
3.3 Considerations.....	35
3.4 Granularity	36
3.4.1 Discrete Curve Evolution	37
4. Shape Matching	39
4.1 Introduction	39
4.2 Matching Process For Our Scheme	40
4.2.1 Reference Points Matching Algorithm.....	40
4.2.2 Partial Matching Procedure	42
4.2.3 Matching of Shapes With Different Levels of Details	46

5. Implementation and Results	49
5.1 Software Application	49
5.2 Construction of Qualitative Descriptions	51
5.2.1 Construction of Curve's Description	51
5.2.2 Construction of Non-Curve Vertex's Description	55
6. Conclusion and Future Work	58
6. Conclusion	58
6.2 Future Work	60
Bibliography	61
Appendix A	65
Appendix B	76

LIST OF FIGURES

<i>Figure</i>	<i>Page</i>
2.1 Two RCC relations used in Cohn’s approach.....	5
2.2 Concavities of a shape	6
2.3 Five Contour Codons	7
2.4 A shape described as codons $1^+ 2 1^+ 0^- 1^-$	8
2.5 A shape divided in curvature extremum	9
2.6 Seven qualitative curvature types.....	10
2.7 A shape represented by seven curvature types	10
2.8 Transformation of a shape outline.....	11
2.9 Distinct example of outline type $\succ\prec\prec\prec$	12
2.10(a) two point orientation grid (b) A tripartite line polygon.....	13
2.11 12 distinguishable arrangements.....	14
2.12 shapes represented by tripartite line tracks (TLT) scheme.....	14
2.13 Indistinguishable shapes represented by tripartite line tracks (TLT).....	15
2.14 Determining angle of a reference point	16
2.15 Determining convexity of a reference point	17
2.16 Three points and oriented line.....	17
2.17 Two distances from center point c	18
2.18 Three curvature types	19
2.19 Complete Shape description using reference points.....	19
2.20 (a) 10 degree acute (b) 85 degree acute (c) 45 degree acute.....	22
2.21 (a) $V_j = [\text{convex, acute, equal}]$ (b) $V_j = [\text{convex, acute, equal}]$	23
2.22 Variations of length within qualitative measure bigger	23
2.23 Two different shapes with same description.....	24
2.24 Example of Mosaic Tiles	25
2.25 (a) both curves are acute (b) both curves are plane	25

3.1 Seven qualitative angles descriptions	28
3.2 A simple shape	29
3.3 Description of relative length.....	32
3.4 Description of equal lengths, $L_j = [\text{equal-bigger } 1x+]$, where $a=b$	32
3.5 curve $j = [\text{curve, convex, acute, smaller } 2x+]$	33
3.6 Complete Description of a shape.....	34
3.7 Shapes that contains constraints of this scheme.....	36
3.8 A few stages of DCE.....	37
4.1 Partial matching example, small shape is part of big shape.....	44
4.2 Partial matching example, two shapes have common subparts	45
4.3 Two shapes have different levels of details	47
4.4 Matching results of two shapes at different levels of details	48
5.1 Snapshot of software application	50
5.2 Snapshot of drawing tool.....	51
5.3 Triangle IJK formed by three vertices I, j and k.....	56

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to Dr. Thomas Barkowsky for his supervision and guidance throughout this thesis. I greatly appreciate the guidance and time that Thomas has given me over the period of this thesis. Senior thanks to Kerstin Schill for reviewing this thesis as second supervisor. In addition, special thanks to Sven Bertel who has been taking part in meetings and discussions. I really appreciate his suggestions and ideas which helped me to reach the completion of this thesis. Thanks to Uwe Schindler of PANGAEA, Bremen for sharing geographic data of countries boundaries.

Thanks to my wife who has always been supporting me emotionally throughout this thesis. Finally, I want to thank to my father Ch. Khursheed Ahmed for providing me this education and supporting me for so many years.

In the name of Allah, the Gracious, Ever Merciful

CHAPTER 1

1. INTRODUCTION

Our vision system encounters objects and their shapes in our environment, recognizes them and represents them in some form in our memory. This information is later retrieved to reconstruct or compare with other objects (and their shapes). Shape is a ubiquitous and significant spatial entity used in our daily life reasoning and intelligent actions. Therefore, recognizing and manipulating these shapes is an important phenomenon to reflect an intelligent behavior. However, shape is a complex spatial attribute because of its almost infinite variation and richness. Consequently, it is important to develop shape representation schemes for computers in order to make them reflect same intelligent behavior as human do.

Representation of shapes is an important concern of today's research in a wide range of areas. There are many applications that can be thought to be benefited from qualitative shape representation schemes including Geographical information system (GIS), robotic navigation, high level visual processing, engineering design, common sense reasoning about physical systems and specifying visual language and semantics.

Objects can be described in terms of their boundaries (boundary makes a shape), where boundary is an infinite set of points that can be quantitatively represented in forms of numerical functions. Some shapes are more complex for them it is generally difficult to find a numerical function for describing their boundary. In those cases piecewise interpolation methods can be used, a set of functions then

make up the quantitative representation of the shape. Different levels of processing is involved doing this task starting from getting sensory data as input and extracting information at a higher level. At first, it is necessary to process the data at lower level to extract segments and surfaces that are related to lower level computer vision (see details, [Ballard & Brown 1982, Boyle & Thomas 1988]). For higher level visual processing, some qualitative representational models have been derived from primal information of previous levels for example qualitative model of stick_gurepresentation [Marr & Nishihara 1978] and componential approach of [Biederman 1987]. In this thesis we are interested in qualitative representation of shapes and will be focusing on geographical shapes found in geographical information systems.

Geographical information systems (GIS) involve a large amount of shape related data where matching of spatial information is one of the major tasks that are required to perform frequently. Moreover, the spatial input may be diverse in nature, i.e. spatial input may exist at different levels of granularity than the information stored in geographic database or spatial input may not be complete. Such complicated spatial matching tasks which are more qualitative in nature require efficient shape representation approaches that support to perform these tasks. In this thesis, we choose to focus our attention to develop a representation scheme that supports to solve these complicated spatial tasks.

1.2 THESIS STATEMENT

A research investigation to provide a qualitative shape representation scheme in which the acquired representations are distinguishable for shapes which are cognitively distinguishable. And to implement matching for acquired qualitative representations of shapes. Matching supports two types of matching i.e. partial matching and matching of shapes at different level of details.

Schemes for representing two dimensional shapes have a major issue that they cannot distinguish some cognitively distinguishable shapes within their scope of shapes. In this thesis we address this issue and adopt an existing scheme [Museros, L. and Escig, M. T. 2003], which already stands out with respect to this problem but still it does not completely overcome the issue, extend it to address this issue at a cognitively satisfaction level considering more positional and relative length information in the representation. Furthermore, in this thesis we also deal with matching of shapes and our extension specifically supports shape matching specifically partial matching and matching of shapes at different levels of details.

Goal of this research is to find a new approach that is close to the way human recognize the shapes and processes shapes in daily lives. The new approach should be implementable and support the computer vision tasks like object recognitions and matching.

CHAPTER 2

2. QUALITATIVE SHAPE REPRESENTATION, STATE OF THE ART

Qualitative representation schemes are most useful for applications of AI that are concerned with the identification and classification of objects that involve reasoning of shapes. Qualitative shape descriptions focus on the relatively high level of shape qualities that are considered most important within a particular context, including how many concavities a shape has, whether it has any lines of symmetry, is it rectilinear or curvilinear, or does it have any holes, etc [Galton and Meathrel 1999]. In a broader view, the origin of qualitative descriptions is likely that some processes are carried out to derive a high level description based on low level inputs. For example, in computer vision, low level raster images are commonly the inputs and need to be converted in higher level representation before doing object identification.

Broadly looking at existing approaches of qualitative shape representation, there exist two main categories, volume or region based approaches and outline or boundary based approaches. In regard to first category, a volume based approach, 3D model representation using simple cylindrical primitives, was developed by [Marr and Nishihara 1978]. Another volume based approach, known as *geon-based* representation, was developed by [Biederman 1987]. Biederman described that a complex solid object can be described with a number of smaller solid pieces put together in a particular manner, for example an arc connected to a cylinder can represent a cup. On the other hand, the second category, boundary based approaches, rely more on 2D domain. These approaches characterize the form of outline of shapes considering the variation in the curvature of its boundary across different positions. In this category there exist approaches of [Hoffman and

Richards 1982], [Layton 1988] and more recently qualitative outline theory of [Galton and Meathrel 1999] , tripartite line tracks of [Gottfried 2002] and reference points based shape representation by[Museros, L. and Teresa, M. 2003] are significant.

In this session we will describe some existing approaches of qualitative shape description in 2D domain.

2.1 COHN'S REPRESENTATION

[Cohn 1995] developed a region based approach to describe the shape, inspiring from logical theory of space called Region Connection Calculus (RCC). Cohn selected two RCC relations for describing relations in its representation scheme. See Figure 2.1.


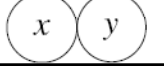
	<i>Relation</i>	<i>Meaning</i>
	$DC(x, y)$	x is disconnected from y
	$EC(x, y)$	x is externally connected with y

Figure 2.1 Two RCC relations used in Cohn's approach

A shape is described by number of concavities that a shape has and with the regional connection relation that hold between each pair of concavities. First step is to find concavities of the shape that correspond to maximal connected parts of geometric inside of the shape. Geometric inside of a shape is given by the remainder of the convex hull of the shape minus the region occupied by the

shape itself. The Figure 2.2 has concavities c_1 , c_2 , c_3 and c_4 . Each pair of concavities is either disconnected (DC) or externally connected (EC). Figure 2.2 c_1 and c_2 has EC relation, while the rest of the pairs of concavities have DC relation. Therefore, basic description of shape using these relations would be as follows:

$$EC(c_1, c_2) \wedge DC(c_1, c_3) \wedge DC(c_1, c_4) \wedge DC(c_2, c_3) \wedge DC(c_2, c_4) \wedge DC(c_3, c_4)$$

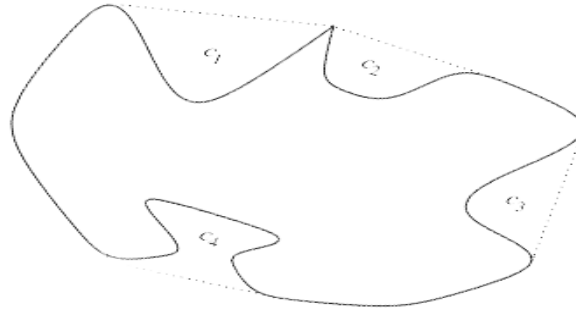


Figure 2.2 Concavities of a shape

Cohn then introduced partitioning technique as a supplement to this technique so that the description of different shapes that hold same relations can be distinguished. In this context four predicates are introduced: $Adjacent(c, c')$, $SemiSide(c, c')$, $SsColinear(c, c')$ and $SsNotColinear(c, c')$. $Adjacent(c, c')$ holds when concavities c and c' are adjacent to one another around the perimeter of a shape. $SemiSide(c, c')$ holds if concavities c and c' occur on the same side of a shape. $SsColinear(c, c')$ holds if a straight line can be drawn between all of EC arms of concavities. $SsNotColinear(c, c')$ holds when $SameSide$ holds but $SsColinear$ does not. Applying these predicates to the pervious description of shape can now distinguish between different shapes that hold same relations between concavities and are also adjacent.

2.2 CONTOUR CODONS

[Hoffman & Richards 1982 and 1984] introduced contour codons technique for describing smooth planer curves for recognition purpose. A contour codon is piece of a curve that is circumscribed by curvature minima and that can contain either zero, one or two points of zero curvature. There are five codon types are defined by Hoffman and Richards i.e. 0^- , 0^+ , 1^- , 1^+ and 2 , reflecting how many number of zero curvature points present in segment of curve. 1^- and 1^+ both contain one point of zero curvature but the difference is, for 1^- point of zero curvature occurs before the point of maximum curvature whereas for 1^+ point of zero curvature occurs after the point of maximum curvature.

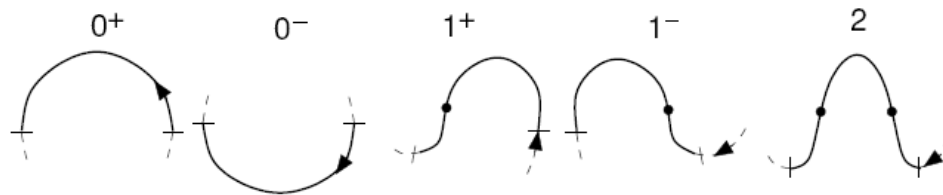


Figure 2.3 Five Contour Codons

A curve is segmented into parts according to the presence of negative curvature minima. An example description of a shape is illustrated in Figure 2.4. Both types of shapes, open and closed curves, can be represented by codon strings. Richards and Hoffman described certain constraints for joining together a pair of codons smoothly. Table below shows those constrains.

	0 ⁺	0 ⁻	1 ⁺	1 ⁻	2
0 ⁺	X		X		
0 ⁻		X		X	X
1 ⁺		X		X	X
1 ⁻	X		X		
2		X		X	X

Table 2.1 Allowed smooth joins for a pair of codons

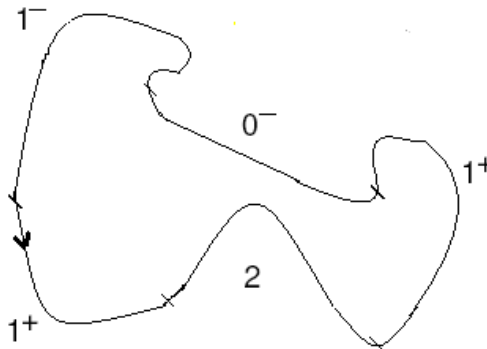


Figure 2.4 A shape described as codons 1⁺ 2 1⁺ 0⁻ 1⁻

Constraints mentioned above limits the scope of shapes that can be represented by this scheme with great extent. For example number of possible combinations for length 6 is $5^6=15,625$, while number of possible legitimate open and closed strings are 643 (calculated through a formula given by Richard and Hoffman).

[Rosin 1993] further extended this approach in the context of multi-scale representation and matching of curves. He supplemented a large number of codons with originals five codons to support the description of curves that contains straight line segments and tangent discontinuities. Rosin's extension approach still cannot distinguish between circles and ellipses.

2.4 QUALITATIVE OUTLINE THEORY

A qualitative boundary based shape representation scheme was developed by Antony Galton and Richard Meathrel in 1999. They describe outline of a shape qualitatively by seven curvature type alphabets see the figure 2.6. Consequently a shape can be qualitatively represented by a string of these curvature type alphabets under some specified grammar rules, see figure 2.7.

- / Straight line segment
- ⊃ Convex curve segment
- ⊂ Concave curve segment
- > Outward pointing angle
- < Inward pointing angle
- ⋈ Outward pointing cusp
- ⋈ Inward pointing cusp

Figure 2.6 Seven qualitative curvature types

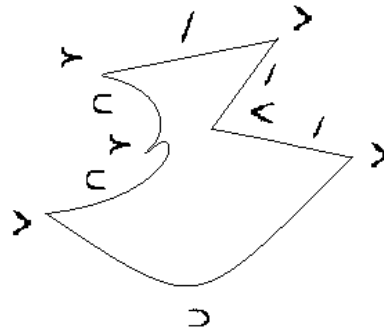


Figure 2.7 A shape represented by seven curvature types

Galton and Meathrel has presented some constrains of this scheme for example, a shape must not contain two consecutive occurrences of the same curvature-type symbol etc. therefore they have made a formal grammar for the scheme which generates all the shapes which satisfy the constraints. This makes scope of the representation scheme limited. In this scheme authors has also presented a

mechanism to transform the representation of a shape to different granularity level. Since the idea of granularity has received a considerable attention in the spatial knowledge representation community therefore this aspect of the scheme makes it more popular. It describes that cusps look very much like angles from a distance therefore cusps ($\succ \prec$) can be converted to angles ($> <$) in order to transforming the representation to coarser granularity level. Moreover authors also present the smoothing and merging transformations on a given representation, see figure 2.8.

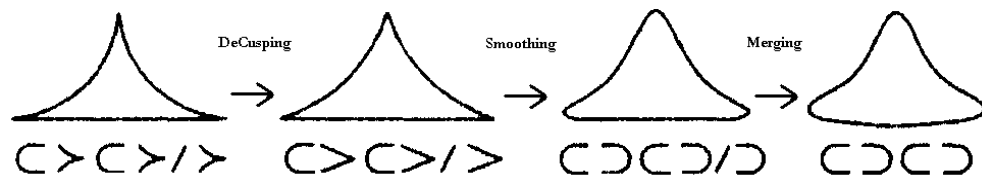


Figure 2.8 Transformation of a shape outline, adopted from [Galton and Meathrel 1999, Fig.2]

[Meathrel and Galton 2000] proposed further enhancement in their scheme in which they obtain set of atomic tokens, based on tangent bearing and curvature of shapes, from which higher level tokens can be derived.

Galton and Meathrel approach has some limitations, as it is already discussed above that the constraints given by this scheme are limiting the scope of the shapes that can be represented with this scheme. Moreover, another limitation that is shown by authors by themselves [Galton and Meathrel 1999] is shown in figure 2.9. Five shapes are shown in figure 2.9 which are indistinguishable by their approach and even by the enhancements they had proposed [Meathrel and Galton 2000] and [Meathrel and Galton 2001]. However they have proposed two ways in which their approach can be extended so that it will be able to distinguish

the shapes in figure2.9. Firstly, each symbol should be annotated by indices i.e. L, R, U, D for left pointing, right pointing, upwards pointing and downwards pointing respectively. [Gottfried 2005] Criticize this idea by saying, “Unfortunately, this means their approach ceases to be rotation invariant; an important property when using such a shape description for indexing image database with arbitrarily aligned objects.”

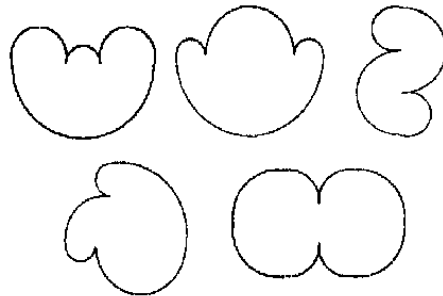


Figure 2.9 Distinct example of outline type $\supset \leftarrow \supset \leftarrow$.
 [Galton and Meathrel 1999, Fig.4 p.5]

Secondly, the line like symbols (\supset in the example) could be annotated by indices denoting their relative lengths. This is however a conspicuous possibility that has also been suggested by some other researchers (such as [Jungert 1993]. [B. Gottfried 2005] Again criticize on the second possibility of measuring the relative lengths as it involves quantitative operations. However, Meathrel and Galton’s idea is not completely quantitative because they actually suggested measuring the relative lengths and representing them in qualitative form such as S (for short), M (for medium) and L (for long).

2.5 TRIPARTITE LINE TRACKS REPRESENTATION

Another approach of qualitative shape representation, based on tripartite line tracks (ILT), a set of qualitatively distinct polygons, is developed by [B. Gottfried 2002]. This approach used reference points orientation grid, originally introduced by [Freksa and Zimmermann 1992], to create a set of 36 different tripartite line polygons. In figure 2.10a orientation grid is shown where two dark points are two reference points and three dotted lines are reference lines. In figure 2.10b an example of tripartite line polygon is shown. These three connected lines can be arranged in many different ways which can be perceived qualitatively different.

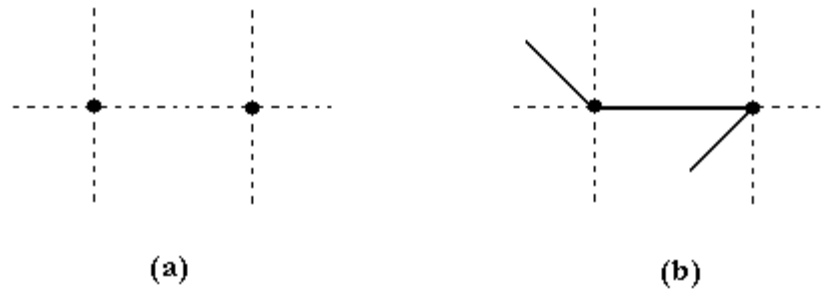


Figure 2.10(a) two point orientation grid (b) A tripartite line polygon

Gottfried further reduce the number of different line track arrangements from 36 to only 12 by removing all symmetrical cases. Figure 2.11 shows these 12 distinguishable arrangements, their naming convention (C_i e.g. C_{12}) is with respect to the set of 36 line tracks.

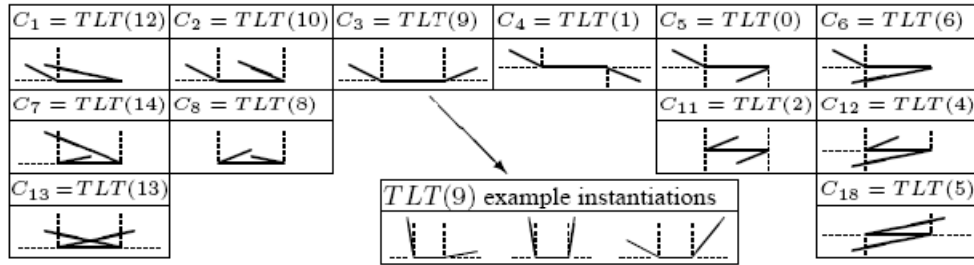


Figure 2.11 12 distinguishable arrangements, figure taken from [Gottfried 2002, Fig. 3]

Figure 2.12 shows examples of qualitative representation of four shapes using tripartite line tracks scheme, where numbers (e.g. 10, 10, 10, 10) represent identification of 12 distinguishable arrangements shown in figure 2.11.

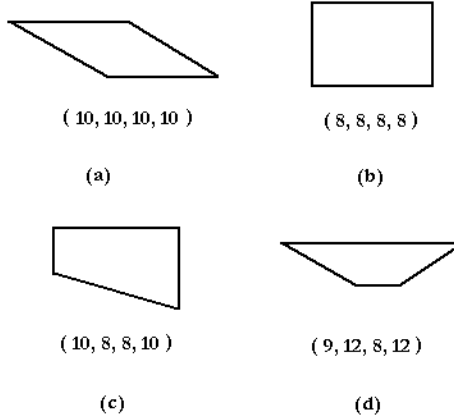


Figure 2.12 shapes represented by tripartite line tracks (TLT) scheme, figure adopted from [Gottfried 2002, Fig. 7]

There are two limitations which are obvious in this scheme; first, it cannot represent curves and circular shapes. Secondly, each TLT arrangement can vary

significantly at its end points within same 'TLT' concept. For example in figure 2.13 all the shapes are represented by same 'TLT' concept. Therefore 'TLT' scheme cannot distinguish between shapes in figure 2.13.

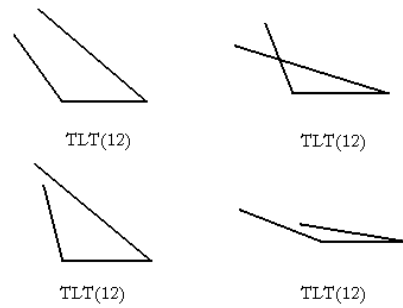


Figure 2.13 in distinguishable shapes represented by tripartite line tracks (TLT), figure adopted from [Gottfried 2002, Fig. 9]

However, Gottfried has also suggested two enhancements to overcome some of its limitations. Firstly, further positional information should be considered to obtain improved results from this approach. Secondly, relative length information of could be taken into account of TLTs.

2.6 REFERENCE POINTS BASED SCHEME

Museros and Escrig's scheme uses reference points and qualitatively describes features of each reference point considering angles, relative side length, concavities and convexities, and types of curvature of the boundary. Where reference points mean the points which completely specify the boundary of a shape for instance, for a polygonal boundary the vertices of the polygon are used as reference points. However, reference points for circular shapes and curvilinear

segments of shapes are a starting and ending point of curve and a point of its maximal curvature. Three qualitative features are determined for a reference point which differs if the reference point is taken from a straight segment or from a curvilinear segment. For straight segments three features are A_j , C_j and L_j , where A_j means the angle for the reference point j , C_j means the type of convexity of reference point j and L_j means the relative length of two adjacent edges. For curvilinear segments three features are ‘curve’, C_j and TC_j , where symbol ‘curve’ indicates it is a curve segment, C_j describes the type of convexity for point j and TC_j describes the curvature type of the curve associated to the point j .

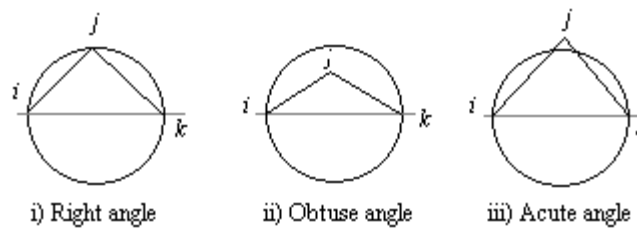


Figure 2.14 Determining angle of a reference point

Let us assume j is a reference vertex and i , previous vertex and k , following vertex are the two relevant vertices of j to determine its qualitative description. Then angle of a reference vertex j is determined by placing an oriented line from point i to k and a circle of diameter equal to the length of oriented line is drawn as shown in figure 2.14. If the reference point j lies on the boundary of the circle then the angle of the point j is right angle. If j exists in the exterior of the circle then the angle of j is acute. And if j exists in the interior of the circle then the angle of j is obtuse, see figure 2.14. Convexity of the reference point j is determined by drawing an oriented line from i to k as shown in figure 2.15. If the reference point j exists on the left side of the oriented line then the convexity of

point j is convex and if point j exists on right side of the oriented line then the convexity of j is concave. Similarly relative length of j is determined by comparing the two lengths, from i to j and from j to k . One of the three qualitative labels (smaller, equal, and bigger) is assigned to j by comparing if length \overline{ij} is smaller, equal or bigger from length \overline{jk} respectively.

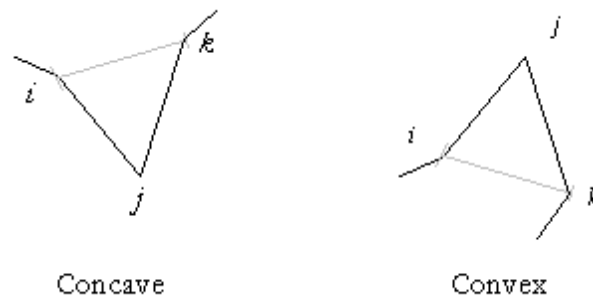


Figure 2.15 Determining convexity of a reference point

To describe curvilinear segments of shapes, first it is determined that if a shape has a curvilinear segment or not, a symbol *curve* in the description, is used to indicate a curvilinear segment. Then three relevant points are used to describe curvilinear segments they are starting and ending point of curve and the point of maximal curvature of curve, see figure 2.16. However, the description is only associated to the point of maximal curvature.

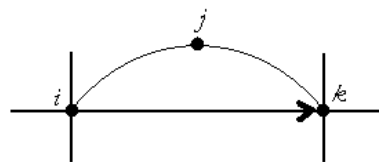


Figure 2.16 Three points and oriented line

For a curve let i is the start of the curve, k is end of the curve and j is the point of maximal curvature. Then convexity of the point j is determined by placing an oriented line from point i to k , if the point j remains on the left side of the oriented line then point j is convex vertex and point j is concave if it remains on the right side of the oriented line. Point j is the point of maximal curvature therefore it is not possible that it remains on the oriented line.

Next step in the description of curvilinear segments is to determine the type of curvature. Type of curvature is determined by comparing two distances d_a and d_b as shown in figure 2.17. A center point c of oriented line \overline{ik} is calculated. First distance (d_a) is calculated from point i to center point c and second distance (d_b) is calculated from point j to center point c .

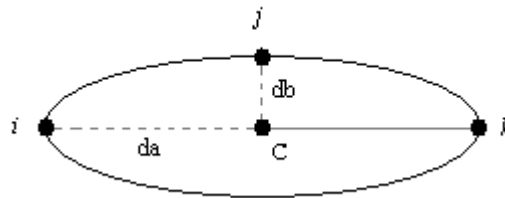


Figure 2.17 Two distances from center point c

Then both distances are compared; if d_a is less than d_b then curvature type of point j is acute, if both d_a and d_b are equal then curvature type of j is semicircle and if d_a is less than d_b then curvature type of j is plane, see figure 2.18.

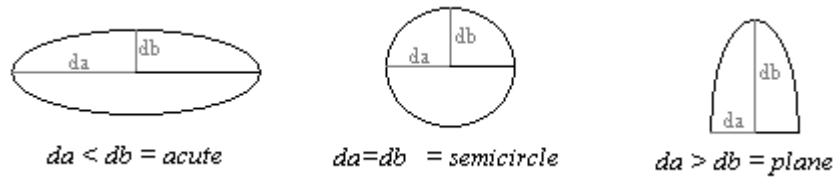


Figure 2.18 Three curvature types

This scheme also deals with the objects with holes and specifies a formalism to qualitatively describe holes in their representation. Since we are not concerned with such types of objects in this thesis therefore we leave details related to the description of holes in an object. The process of giving the qualitative description has to be repeated for each vertex of the boundary for complete description of a shape. Then for a given shape (shape without holes) its complete description would be following:

[holes_type, curve_type, [Colour, [A1, C1, L1 | curve, C1, TC1] [An, Cn, Ln | curve, Cn, TCn]]

Where n is the number of vertices (reference points) of the boundary of a shape. The 'hole_type' belongs to the set [without-holes, with-holes], the curve_type belongs to the set [without-curves, with-curves, only-curves]. A_i , C_i , L_i are qualitative angle, convexity type and relative lengths of consecutive vertices to the reference vertex. TC_i is the qualitative description of curvature type of curvilinear segments. Figure 2.19 shows a shape and its description using this scheme.

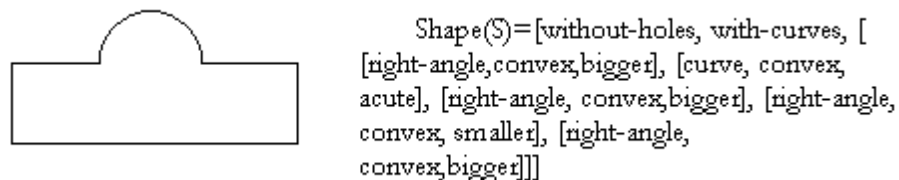


Figure 2.19 Complete Shape description using reference points

2.7 DISCUSSION

In this chapter we have presented some existing qualitative representation schemes given in scientific literature. A remark from Cohen and Hazarika is worth coating here as an overview of existing approaches, “It is unlikely that single universal spatial representational language will emerge rather, the best we can hope for is that the field will develop a library or representational and reasoning devices and some criteria for their most successful applications.” [Cohn & Hazarika 2001].

A comparative analysis of qualitative representation schemes described in this chapter shows us that in Cohn’s representation it is not straightforward to describe polygons moreover all convex shapes have same description. Hoffman and Richard’s contour codon Leyton’s initial set of extremum primitives are limited to describing smooth, continuously differentiable, closed curves. Galton and Meathrel’s representation is although not that old and they have considered the limitations of existing schemes and tried to overcome them. However, their own approach cannot distinguish some cognitively distinguishable shapes, see figure 2.9. Galton and Meathrel have also suggested, how to transform their representation to a different of granularity to introduce de-cusping and smoothing and merging operations. But this transformation is very limited and cannot truly transform a complete representation to another level of granularity. Tripartite line track (*TLT*) approach of Gottfried is even latest then Galton and Meathrel’s approach. Gottfried’s approach is not able to describe curvilinear smooth shapes for example circles, ellipses and arcs. Secondly, it also has problem of not distinguishing cognitively distinguishable shapes. As describes in the approach above all the lines which exist in same region of reference grid cannot be distinguished, see figure 2.13.

The question of granularity levels is not seriously addressed in the schemes presented in above section. Only Galton and Meathrel mentioned it explicitly and laid down some rules for limited transformation of their representation at different levels as discussed above. Another recent approach that is based on Gottfried's existing approach, addresses this question in more detail, called scope histogram approach by [Schuldt et al. 2006]. However, this approach puts a limitation that a shape cannot be retrieved from the given representation because it does not preserve sequence information of contours it uses for shape description.

The reference points based qualitative representation approach of Museros and Escrig is most recent therefore appear most promising as well. It can more precisely describe a shape qualitatively considering some relative information with respect to immediate neighboring points within the shape. Therefore the problem of not distinguishing the cognitively distinguishable shapes has been much reduced in this scheme with respect to previously existing schemes. All the existing schemes had been developed for specific requirements and describe limited scope of shapes. However, over the period of times requirements and the scope have been becoming more wide-ranging.

In this thesis being inspired from reference points based scheme of Museros and Escrig we adopt this scheme and extend it from a coarse qualitative description of shapes to a fine qualitative description of shapes. In the following section we will describe and discuss some limitations of reference points based scheme to support our work of extending this scheme.

2.8 LIMITATIONS OF REFERENCE PINTS BASED SCHEME

In state of the art schemes we have presented reference points shape representation scheme of Museros and Escrig. Descriptions of shapes using this scheme sometimes do not distinguish some cognitively distinguishable shapes; this limitation is shared by every scheme in their scope. Whenever a scheme adopts qualitative aspects of a shape to represent the shape qualitatively, it definitely loses much of information of the exact boundary of the shape, which causes the problem of indistinguishability. For example in this scheme, there are only three qualitative angle descriptions that are acute, obtuse and right angle to determine qualitative angle of a reference vertex. Where acute angle varies from $(0^\circ, 90^\circ)$ such that $\{\text{angle} \mid 0^\circ < \text{angle} < 90^\circ\}$ and obtuse angle varies from $(90^\circ, 180^\circ)$ such that $\{\text{angle} \mid 90^\circ < \text{angle} < 180^\circ\}$. This is much of a variation to lie in one cognitive qualitative measure. We humans can easily distinguish between 10, 45 and 85 degree angles, see figure 2.20. On the other hand one of the qualitative descriptions of angles in this scheme is right angle, although, right angle has a cognitive significance but it is purely a quantitative measure. However, 85 degree and 95 degree angles are not easily distinguishable by humans when appear in different orientations in boundaries of shapes, most often they are perceived as right angles. Therefore cognitive distinguishability must be considered in a qualitative shape representation scheme.

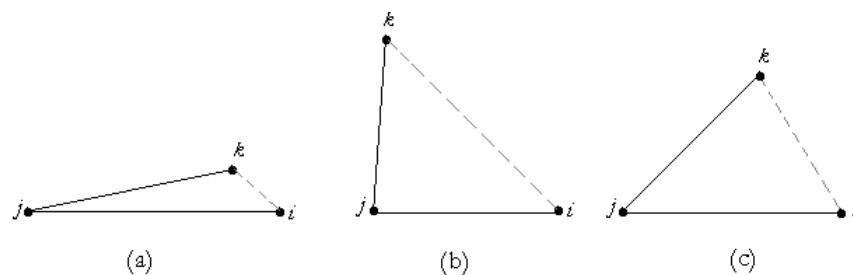


Figure 2.20 (a) 10 degree acute (b) 85 degree acute (c) 45 degree acute

When convexity and angle descriptions do not help to distinguish two distinguishable shapes, then sometimes relative length description does help. But in a case when both lengths are equal and angles differ by a large value lying within the range of acute angle. See figure 2.21 where both reference vertices have exactly same description but they are clearly distinguishable. There are a large number of such shapes which have equal lengths but their acute or obtuse angle differs by a large value.

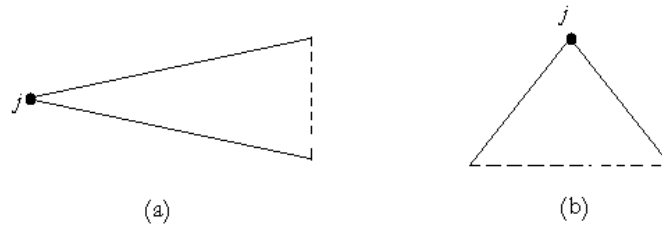


Figure 2.21 (a) $V_j = [\text{convex, acute, equal}]$ (b) $V_j = [\text{convex, acute, equal}]$

When two vertices have same convexity and same angle but different lengths of their two adjacent edges then, in such cases if both vertices also get the same relative length description (smaller or bigger). However the differences between the lengths of two vertices differ by a large extent such that two shapes are clearly distinguishable but still both vertices share the same description, see figure 2.22.

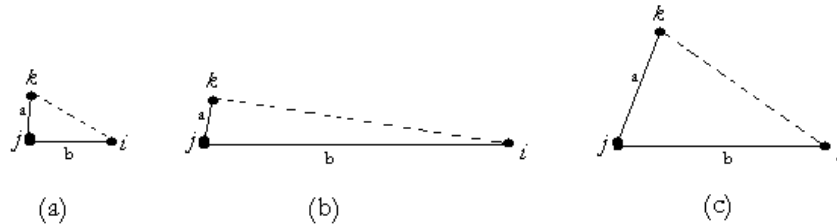


Figure 2.22 (a) $V_j = [\text{convex, acute, bigger}]$ (b) $V_j = [\text{convex, acute, bigger}]$ (c) $V_j = [\text{convex, acute, bigger}]$

Moreover, worst case can be when both ambiguities those of undistinguishable angles and relative lengths appear together in two shapes, and they are declared same by the scheme, see figure 2.23, where two cognitively distinguishable shapes share the same qualitative description with Museros and Escrig's scheme.

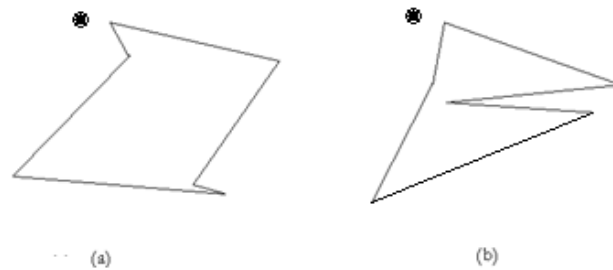


Figure 2.23 Shape (a) and (b) has same description, shape = [[convex, acute, small], [convex, acute, equal], [concave, acute, bigger], [convex, acute, small], [convex, acute, bigger], [concave, obtuse, bigger]]

However, Museros and Escrig have successfully applied this scheme in an industrial application for recognizing mosaic tiles. The above mentioned limitations do not occur in the case of mosaic tiles because those tiles are small closed shapes with few vertices see figure 2.24. All tiles have some kind of symmetry; although they have different shapes but cannot vary too much in size and shape. The concept of holes description on the tiles texture helps a lot to distinguish between two tiles because holes appear in different positions on the tiles. In this thesis we are interested in geographic shapes which have no symmetry and also they can contain large number of vertices and even convexity

of vertices can also vary which further increases the possibility of occurrence of above mentioned limitations when using this scheme.

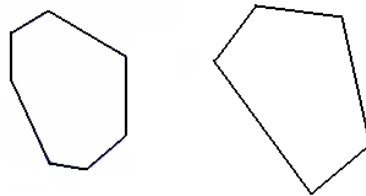


Figure 2.24 Example of mosaic tiles

Similar type of limitations lies in the description of curvilinear segments using this scheme. The scheme presents only three qualitative measures for describing type of curvature of a curve that is plane, semicircle and acute. In these qualitative categories there are so many curves which are perceived different but lies under one of the above qualitative measures, see figure 2.25 as an example. Moreover, in the case of semicircle there is no description of its size to differentiate one from another of different size. This problem is similar to that of non curve segments where both lengths of a reference vertex become equal and they are not distinguished. Relative size information with respect to a neighboring reference points needs to be described in order to distinguish such similar segments which only differ in size.

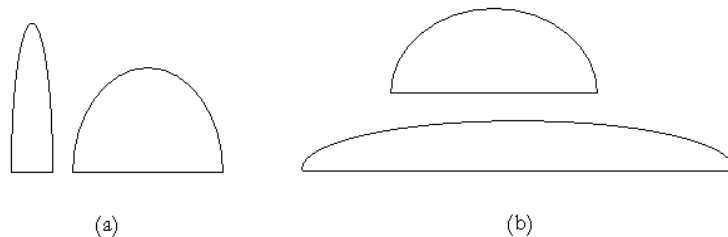


Figure 2.25 (a) both curves are acute (b) both curves are plane

CHAPTER 3

3. EXTENDED REFERENCE POINTS BASED REPRESENTATION

This chapter introduces extensions for this scheme to overcome the limitations and presents complete extended reference points based scheme. In the end Discrete Curve Evolution technique is presented that is used in this thesis for coarsening the shape details.

3.1 EXTENSIONS

We have seen limitations of Museros and Escrig's scheme, in previous chapter, here we will introduce some extensions for this scheme to overcome above mentioned limitations at a cognitively acceptable level. In other words we will extend Museros and Escrig's scheme from coarse qualitative description of shapes to a fine qualitative description of shapes. We will not only extend the existing scheme but we will also make some changes in order to adopt the scheme to our requirements. For example the original scheme describes the color of the shapes and wholes within shapes, both of these features are not required in our requirements therefore we will leave these features in our extended version.

Whenever we attempt to describe any shape in qualitative terms we pay a price of losing the details of exact boundary of the shape. When our vision system encounters a shape it performs some processes on it and stores it in our memory.

At a later time, if we want to reconstruct the same shape, we ask our memory, and it returns some qualitative properties of that shape but it cannot reconstruct the shape exactly. Here a question arises what kind of information can we lose in order to describe a shape qualitatively, or what are the qualitative aspects which are important to be preserved in order to retrieve the shape at a later time. Answers to these questions are not simple but require an in-depth discussion of human cognitive psychology and the sciences of cognition. The purpose of raising this discussion here is to emphasize the point that whatever extensions we make to this scheme or even if we present a totally new scheme, it would still have some limitations. However, we can provide a scheme with certain specification and requirements that can distinguish shapes at a cognitively acceptable level.

Therefore, we introduce three basic extensions that describe qualitative properties of a shape at more detail level than original scheme i.e. Description of angles, relative lengths and curvature type.

3.1.1 DESCRIPTION OF ANGLES

We introduce seven cognitively differentiable qualitative groups of angles shown in figure 3. 1. Qualitative description of an angle of a reference point (reference vertex) is determined by placing an oriented line between previous and following point of the reference point. Triangular geometry is used to find the angle measure of the reference point then one of the respective qualitative angle descriptions is assigned to the reference point according to equations below.

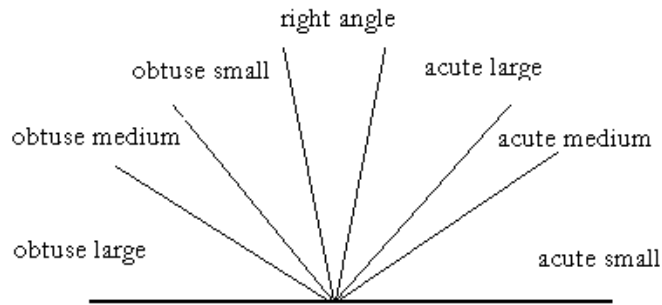


Figure 3.1 Seven qualitative angles

$$acute\ small(x) \Leftrightarrow (0,35) = \{x \mid 0 < x < 35\}$$

$$acute\ medium(x) \Leftrightarrow [35,55) = \{x \mid 35 \leq x < 55\}$$

$$acute\ large(x) \Leftrightarrow [55,80) = \{x \mid 55 \leq x < 80\}$$

$$right\ angle(x) \Leftrightarrow [80,100) = \{x \mid 80 \leq x < 100\}$$

$$obtuse\ small(x) \Leftrightarrow [100,125) = \{x \mid 100 \leq x < 125\}$$

$$obtuse\ medium(x) \Leftrightarrow [125,145) = \{x \mid 125 \leq x < 145\}$$

$$obtuse\ large(x) \Leftrightarrow [145,180) = \{x \mid 145 \leq x < 180\}$$

We have extended the three qualitative angles measures of Museros and Escrig's approach to seven qualitative measures. We support this extension with arguments given in the previous section that only three qualitative angle measures does not qualitatively represent all angles on the scale from zero to 180 degree. This set of seven qualitative angles descriptions is cognitively distinguishable and

close to human spatial psychology. For instance, if a shape given in figure 3.2a is shown to us for five seconds and then we are asked to describe this shape. We will be retrieve the shape in our mind but it may not be the exactly same shape. Our mind reconstructs the shape according to the qualitative properties that it had stored against that shape. For the sake of clarity in the discussion, let us label the shape in figure 3.2a, see 3.2b.

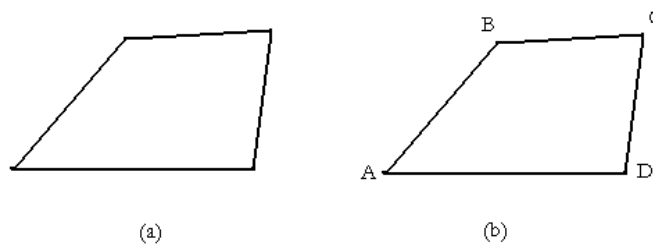


Figure 3.2 A simple shape

Our cognitive system normalizes the angles in qualitative terms, for example angle D and C would most probably be reconstructed as right angles. In the figure above due to the digitization effect we can more easily distinguish if it is right angle or not but imagine this shape drawn on a paper. There it would be difficult to recognize that if they are exact right angles or not. Similarly, angle A would most probably be reconstructed an angle close to 45 degree. This effect shows us that our vision system normalizes angles to a nearby qualitative group of angles. 45 degree, 90 degree and 135 degree angles have a special significance in special psychology. Therefore, we have made qualitative groups around these angles; as a result we get seven qualitative groups in total.

3.1.2 DESCRIPTION OF LENGTHS

In section 2.8 we have discussed the limitation of relative length description regarding Museros and Escrig's approach that qualitative terms 'smaller' and 'bigger' are not enough to cognitively distinguish different lengths of a reference vertex. We further argue that the qualitative length that we store in our mind is more relative to its environment. And the quality of 'how much' is also considered by human while describing a length. Therefore we also consider this quality in our description of lengths.

We extend the relative length description of Museros and Escrig's scheme and introduce a feature 'how many times' in the description. This new feature is inserted after qualitative symbols 'smaller' and 'bigger' in the description of relative length for a reference vertex. Therefore, for a reference vertex j we compare length a , from i to j , with length b , from j to k . Two lengths are compared to answer the question of how many times a relative length is smaller or bigger, see figure 3.3. We compare length a and length b according to a rule that length b would be one time of length a if length b is greater than half of length a and less than one and half times of length a . we have introduced these qualitative intervals for describing how many times feature for length comparison. Since it is not cognitively feasible to compare two lengths which differ more than five or six times with respect to each other, therefore we also consider this fact in our description of relative lengths and state that when relative length is bigger/smaller more than five times it is described as very big or very small. See figure 3.3b where length a is bigger (more than two times of b) than length b . Here two times does not mean exactly two times rather this is a qualitative term that means length b is not exactly two times greater but in an interval of plus/minus half of smaller length. Complete algorithm for determining relative length of reference vertex is given below:

if $Length(a) + 10\% Length(a) < Length(a) \ \& \ Length(a) - 10\% Length(a) > Length(a)$ *then* $RelativeLength(j) = equal$

if $Length(a) < Length(b)$ *then*

$$times = \frac{length(b)}{length(a)}$$

$$remainder = remainder\left(\frac{Length(b)}{length(a)}\right)$$

if $times > 5$ *then*

$RelativeLength(j) = very\ small$

else

if $remainder \leq 50\% Length(a)$ *then*

$RelativeLength(j) = smaller \ \& \ times$

else

$RelativeLength(j) = smaller \ \& \ times + 1$

if $Length(a) > Length(b)$ *then*

$$times = \frac{length(a)}{length(b)}$$

$$remainder = remainder\left(\frac{Length(a)}{length(b)}\right)$$

if $times > 5$ *then*

$RelativeLength(j) = very\ big$

else

if $remainder \leq 50\% Length(b)$ *then*

$RelativeLength(j) = bigger \ \& \ times$

else

$RelativeLength(j) = bigger \ \& \ times + 1$



Figure 3.3 (a) $L_j = [\text{smaller}2x]$ (b) $L_j = [\text{bigger}2x]$

When both lengths (a and b) becomes equal then we compare length a with the oriented line from i to k and apply the same rule of comparison as described above. Result of new comparison is just extended with the ‘equal’ symbol as shown in figure 3.4.

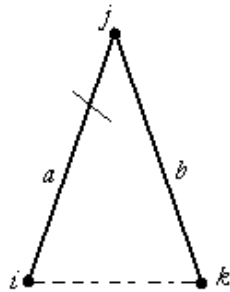


Figure 3.4 Description of equal lengths, $L_j = [\text{equal-bigger } 1x]$, where $a=b$

3.1.3 DESCRIPTION OF CURVATURE TYPE FOR CURVILINEAR SEGMENTS

Reference point based scheme use three relevant points (starting point of curve, maximal curvature point and ending point of curve) to determine the description of curvilinear segments and describe three attributes of a curve segment those are; symbol *curve*, convexity and curvature type. In the extension of this we

introduce fourth attribute that is relative length of two distances d_a and d_b , shown in figure 3.5, in the description of curvilinear segments. Where d_a is the distance from starting point of curve to point c , middle point of oriented line \overline{ik} , and d_b is the distance from maximal curvature point to point c . We use same concept of comparing two lengths that we used in the case of non curvilinear segments. Description of this fourth attribute would make it possible to distinguish between two acute curves or to obtuse curves which vary in curvature.

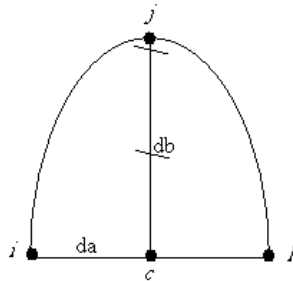


Figure 3.5 curve $j = [\text{curve, convex, acute, smaller } 2x]$

3.2 COMPLETE DESCRIPTION OF SHAPES

We use original reference points based scheme with our extensions discussed in above sections for complete description of a shape. For description of a shape the process of giving the qualitative descriptions has to be repeated for each vertex of the boundary of the shape.

For a given shape its complete description would be as following: $[\text{curve_type, [A1, C1, RL1 | curve, C1, TC1, CL1] [An, Cn, RLn | curve, Cn, TCn, CL1]]$, where n is the number of vertices (reference points) of the boundary of the shape. Symbol curve_type belongs to the set $[\text{without-curves, with-curves,}$

3.3 CONSIDERATIONS

A potential consequence of qualitative description of curvature type of shapes is that two shapes can exist which are cognitively distinguishable but have same qualitative description. We have seen this problem with all existing schemes as discussed in previous chapters. In our scheme we have tried to overcome this problem to a cognitively acceptable level. But there would still be shapes that, while looking different from one another, still share the same description. The reason is not the same as previous but there is another factor involved that is the limitations of local features description. The discussion of local verses global features has been raised by [Gottfried 2005] where he describes that local feature schemes generally consider the ordering of features around the boundary of a shape however there are other properties which are also important, whether two features are directly adjacent to each other, and if not, what kinds of features exist between them e.g. their regional position and orientation. He suggests that such non-local relations have to be taken into account in order to distinguish shapes which are undistinguishable by local feature schemes.

In our scheme we consider relationships of two neighboring features of each vertex which provide some information outside of localness. Therefore our scheme cannot be called a local features scheme but it is mix of local and non-local features. Moreover this scheme can be further extended to include global features in the description.

Our scheme has some limitations for some curvilinear shapes. We describe curves using quadratic Bezier curves therefore more complex curves cannot be described using this scheme however, a quadratic Bezier equation can describe curves with convex and concave parts but in our scheme we only use quadratic Bezier equation with both positive control points. See figure 3.7 that shows two

shapes which contains constraints, where in shape (b) a line between two curves is indescribable using this scheme.

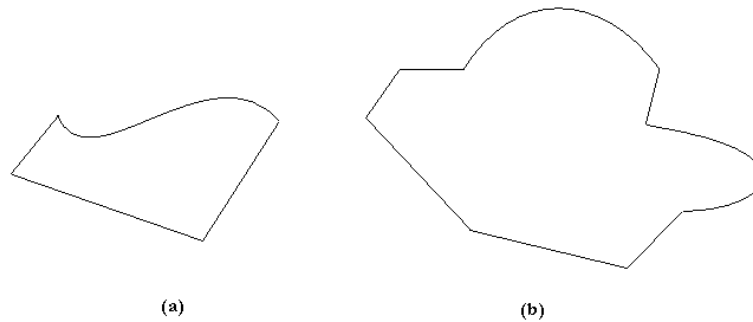


Figure 3.7 Shapes that contains constraints of this scheme

3.4 GRANULARITY

Granularity is the description of components of an object at different levels of details. Granularity means how much information is included in the representation of a shape. It should not be confused with scalability or resolution. For detailed explanation of granularity theory see [Hobbs, J.R. 1985]. It is an important aspect in shape representation and matching. In the context of shape representation, coarse granularity level provides abstract information about the shape whereas fine granularity level provides detailed information about the shape.

In this thesis we want to implement matching of geographical shapes at different levels of detail. When shapes are extracted from maps or other sources they may have digitization noise or they may be at very fine level. Therefore before describing them they must be transformed to a qualitatively acceptable granularity

level. For doing this we use Discrete Curve Evolution (DCE) theory by [Latecki and Lakämper 1999]. However, it is not possible to know, if two shapes are at same granularity level or not, therefore we need to implement matching of shapes with different granularity levels.

3.4.1 DISCRETE CURVE EVOLUTION

An approach of polygonal curves evolution to fit discrete nature of curves, called Discrete Curve Evolution (DCE), was developed by [Latecki and Lakämper 1999]. DCE does coarsening to boundaries of shapes without any blurring effects and distortions of relevant features. When digital images are extracted and segmented they contain missing information on the boundary of the objects due to the digitization and segmentation noise but if it is still possible to recognize overall shape then they miss no information. Latecki and Lakaemper claims that if B is a set of boundary points of an object then there exists a subset A of the set of the boundary points B that is sufficient to represent the shape of the object. Figure 3.8 we can observe this fact.

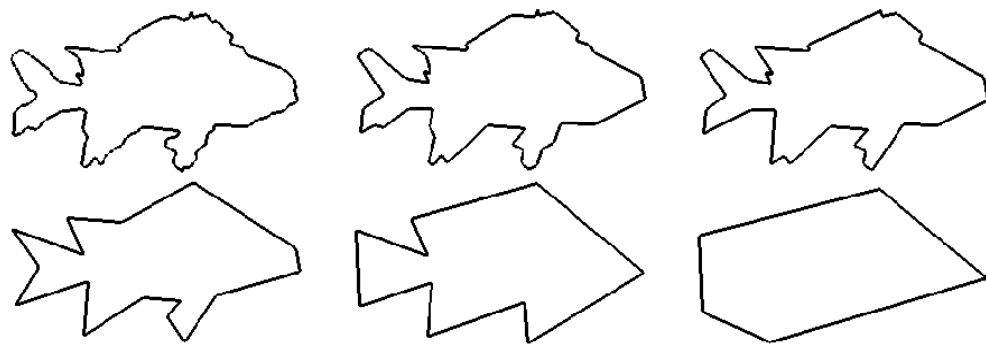


Figure 3.8 A few stages of DCE, figure taken from [Latecki and Lakämper 1999] page.4

The process of DCE calculates the relevance measure, K defined below, of each vertex of the boundary and the vertices whose relevance measure is minimal are deleted from the boundary. As a result DCE process produces a sub set of original boundary points that makes a coarser boundary without losing the relevant information to recognize the shape. However the key factor in DCE is the relevance measure K that is defined by a given formula:

$$K(v, u, w) = K(\beta, l_1, l_2) = \frac{\beta l_1 l_2}{l_1 + l_2},$$

Where v is the vertex for which relevance measure is being calculated, u and w are two neighboring vertices. β is the turn angle at vertex v , l_1 is the length of vu , and l_2 is the length of vw .

The relevance measure K reflects the shape contribution of the vertex v therefore the higher the value of K , the larger is the contribution of arc $vu \cup vw$ to the shape of the object.

CHAPTER 4

4. SHAPE MATCHING

This chapter presents shape matching schemes and procedures used in this thesis. First it provides a brief introduction of shape matching then it describes matching processes used in this thesis. After that, two matching approaches, partial matching of shapes and matching of shapes at different levels of details are presented with their matching procedures and algorithms.

4.1 INTRODUCTION

Shape matching is a fundamental task for object recognition therefore it has an immense importance in visual information systems, computer vision, pattern recognition systems and robotics. Shape matching can be precisely defined as establishing a correspondence between features in one shape and similar features in another shape [Shokoufandeh et al. 2002]. In the applications of shape matching, it involves the task of quickly finding promising contenders from the database on a given query and then checking each contending shape using detailed verification procedure for the best match. Global features shape matching has been popular for a long time, where similarity is measured between entire models. The recognition systems encounter shapes that are affected by digitization and other kinds of noise and also found at different levels of details. Therefore those matching algorithms are needed that can match the shapes which exist at different levels of details. Another problem in shape matching is that there are shapes come across very frequently to the shape recognition systems which are partially hidden or obscured by other objects. Consequently, partial matching and matching of shapes at different levels of details are the concepts

that are gaining great importance these days for many applications. An extensive survey of shape matching can be found at [Veltkamp and Hagedoorn 1999].

This thesis presents a qualitative shape representation scheme and focuses on its application for geographical shapes therefore in this thesis shape matching will be performed in this context. We will present the procedures for matching qualitative representations of shapes obtained from the scheme presented in this thesis. Moreover, we will formalize the procedures for partial matching and matching at different levels of detail.

4.2 MATCHING PROCESS FOR OUR SCHEME

As we have stated before that we implement two types of shape matching in this thesis, matching of shapes that lie at different level of details and partial matching of shapes. In both cases algorithm for comparing reference points stays the same but matching procedure differs. Therefore let us first describe the reference points matching algorithm and then discuss two matching procedures.

4.2.1 REFERENCE POINTS MATCHING ALGORITHM

Given two shapes, let us call them, shape A and shape B, to compare them first their qualitative descriptions are constructed using our extended reference points based scheme. Then for comparing qualitative descriptions of reference points, we use an algorithm known as largest common consecutive sequence (LCS) and extend it for our matching requirements. LCS is generally famous for finding largest common substrings between two strings. Extended LCS algorithm for our scheme is given below:

```

VerticesMatching (Inputs: Set VerticesA, Set VerticesB, Output: Set MatchedVertices)
{
  M = Length[VerticesA] // number of vertices of shape A
  N = Length[VerticesB] // number of vertices of shape B
  Set MatchedVertices

  LCS = Array[M+1, N+1] // Array for finding largest common string

  w = Array[M+1, N+1] // array for consecutive vertices

  for I:=1...M
    for J:= 1...N
      {
        Similarity = CompareTwoVertices ( VerticesA[I-1] , VerticesB[ J - 1] )
        if Similarity ≥ 70)
          {
            K = w[ I-1 , J -1 ]
            LCS [I , J]=LCS[I - 1 , J - 1] + Square(K+1)- Square (K)
            w[I , J] = K+1
          }
        else
          LCS[I , J] = LCS[I - 1 , J - 1]

        if LCS[I - 1 , J] ≥ LCS[I , J]
          {
            LCS[I , J] = LCS[I - 1 , J]
            w[I , J] = 0
          }
        if LCS[I , J- 1] ≥ LCS[I , J]
          {
            LCS[I , J] = LCS[I , J- 1]
            w[I , J] = 0
          }
        if w[i , j] > 0
          MatchedVertices = Add { VerticesA[I -1], VerticesB[J -1]}

      }
    return MatchedVertices
}

```

4.2.2 PARTIAL MATCHING PROCEDURE

Partial matching is matching of sub-parts of a shape with similar parts of other shapes. The sub-parts are the parts that are not predefined and can even be any sub-shape of a large shape. Partial matching is very important because often shapes are encountered which are not complete or obscured by some other objects, therefore in such situations partial matching is very useful. In this thesis we perform partial matching to compare two shapes and search if they have any common sub-parts. Difference of total number of vertices of two shapes is not important for performing partial matching between them. A shape with a small number of vertices can be a sub-part of a shape with a large number of vertices. If two shapes have only one vertex in common then that vertex cannot be called sub-part, however to qualify a common sub-part at least two consecutive vertices of two shapes must be matched. Therefore in our partial matching algorithm we look for consecutive common vertices of two shapes. We use the same algorithm *VerticesMatching* to compare vertices of two shapes because logic for consecutive vertices matching is already in that algorithm. Now we just track the consecutive sub-sequences of vertices from the results of *VerticesMatching* algorithm and get the common sub-parts of two shapes. Algorithm for partial matching is given below.

VerticesMatching algorithm returns a set of matched vertices to which we track their indices to verify that if they are consecutive or not. The matched vertices which do not satisfy the conditions of common sub-part are deleted from the set. Then the remaining vertices are sub-parts of two shapes.

Consecutive common vertices tracking algorithm:

```

i = 0
while(1 < Length [MatchedVertices] -1)
{
    MatchFlage = false
    j = i
    while (MatchedVertices [j++].IndexA== MatchedVertices [i].IndexA+1
            && MatchedVertices [j].IndexB== MatchedVertices [i].IndexB+1)
        {

            if MatchFlage == false
            {
                PartialMatches=Add{ MatchedVertices [i]}
                PartialMatchCount++;
            }
            MatchFlage = true

            PartialMatches=Add{ MatchedVertices [j]}

            if j > Length [MatchedVertices] -1
                break

        }
        i = j
}

```

Let us see some examples to further clarify the concept of partial matching used in this thesis. In figure 4.1 an example of partial matching is presented where two shapes are different in size and in total number of vertices. The shape above in figure 4.1 is a map of Italy and shape below is just a lower part of Italy. Our partial matching algorithm recognizes that the small shape is part of big shape.

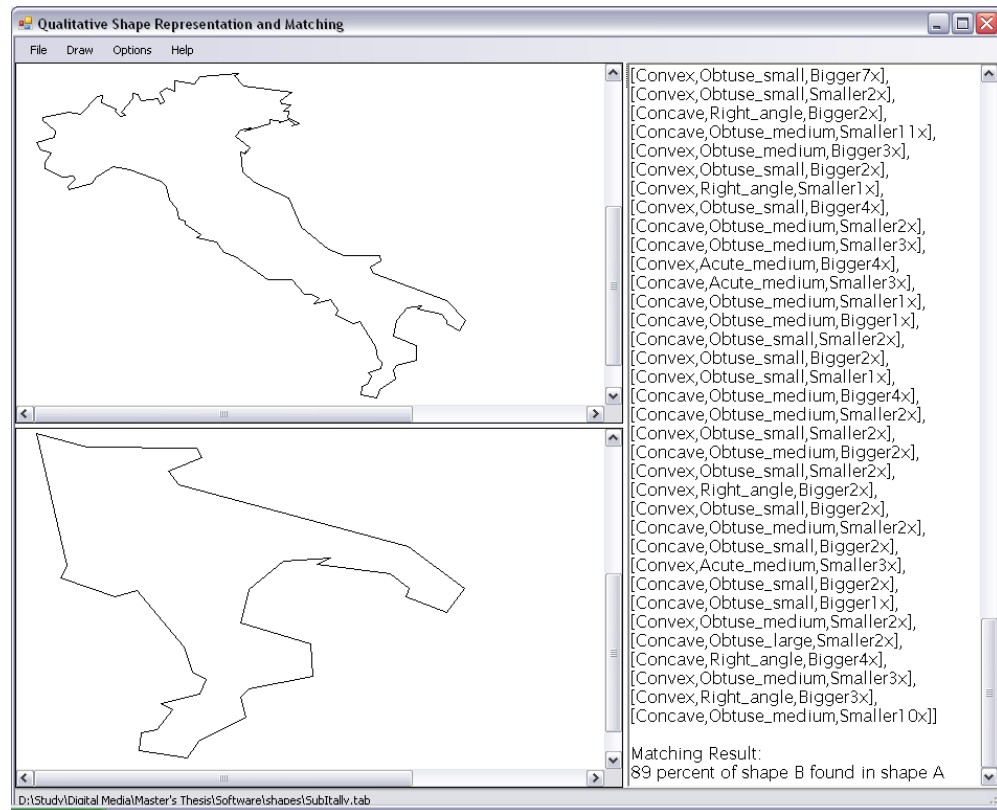


Figure 4.1 Partial Matching Example, small shape is part of big shape

Similarly, figure 4.2 shows another example of partial matching where two shapes have some common parts but are not exactly same. Results of our partial matching algorithm given in figure, '45 % of shape B found in shape A', shows that all common sub-parts of two shapes are successfully recognized.

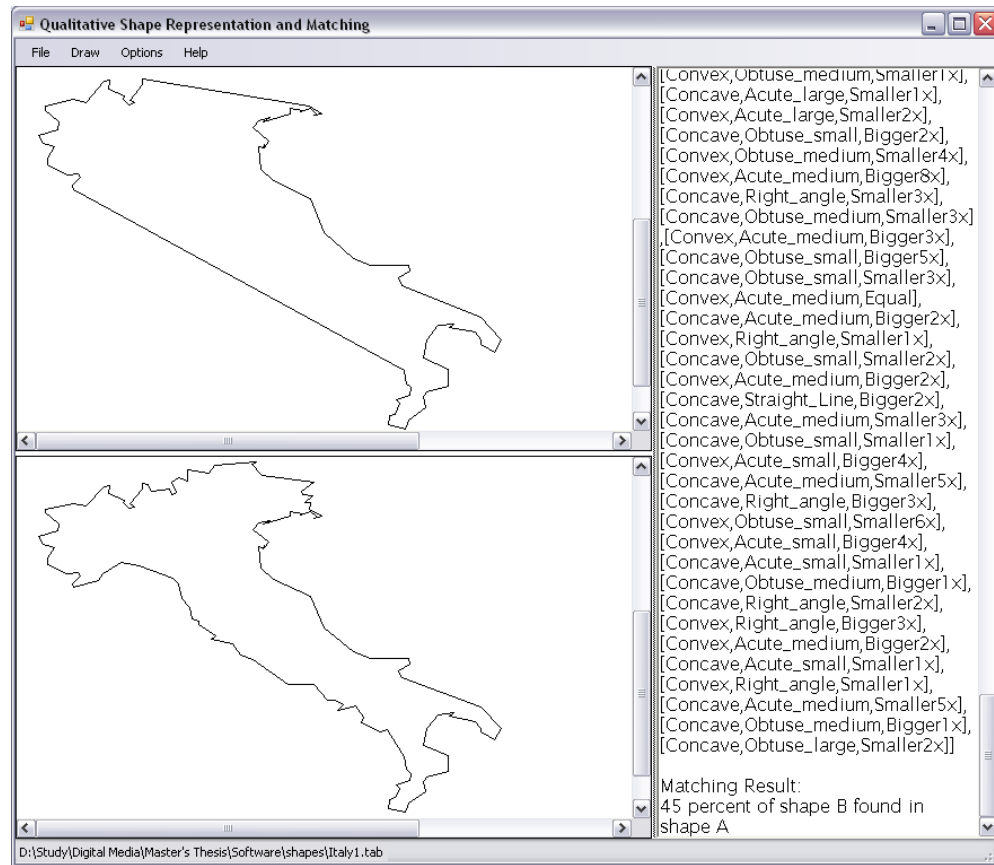


Figure 4.2 Partial Matching Example, Two shapes have common sub-parts

4.2.3 MATCHING OF SHAPES WITH DIFFERENT LEVELS OF DETAILS

When numbers of vertices of two shapes are not equal then there exists a possibility that two shapes might be the same but they were represented at different levels of details. However, they might be two totally different shapes as well, when their numbers of vertices does not match with each other. Here we present our approach for matching such shapes. In this thesis we assume that the system will have a database of qualitative representations of shapes and then a given shape will be compared against the shapes in the database. And we also assume that the represented shapes in database are described at relatively coarser level of details. However, the shapes that are supposed to be given to the system for comparison are prone to be found at fine level of details because they are extracted from images. Therefore, in our assumption we exclude the possibility that a given shape to the system for matching will be at coarser level than the shapes already stored in database of the system. Thus, given a shape with number of vertices more than the shape to be compared with from the database, first DCE algorithm will be applied to transform the shape to coarser level (number of vertices reduces) where the number of vertices of both shapes match with each other (for a given threshold of equality). Then both shapes are compared with *VerticesMatching* algorithm and matching result is analyzed to answer the question that how much percentage of similarity exists between two shapes. See the algorithm below for matching of shapes at different levels of details.

```

if LENGTH[VerticesB] < LENGTH[VerticesA] + Vertices_Thresh
{
    // Matching
    Result = VerticesMatching(VerticesA, VerticesA)
}

```

```
else // when difference of Vertices number is more then thresh hold
{
  while LENGTH[VerticesB] > LENGTH[VerticesA] + Vertices_Thresh
  {
    // Apply DCE
    VerticesB = DCE(VerticesB)
  }
  Result = VerticesMatching(VerticesA, VerticesA)
}
```

Figure 4.3 shows an example of two shapes which are at different levels of details therefore they cannot be matched directly. For example if we want to match shape (b) with shape (a), according to our algorithm shape (b) will be transformed to a coarser level equal to shape (a) using DCE algorithm. Then their qualitative representations will be compared. Figure 4.4 shows the results of our matching algorithm.

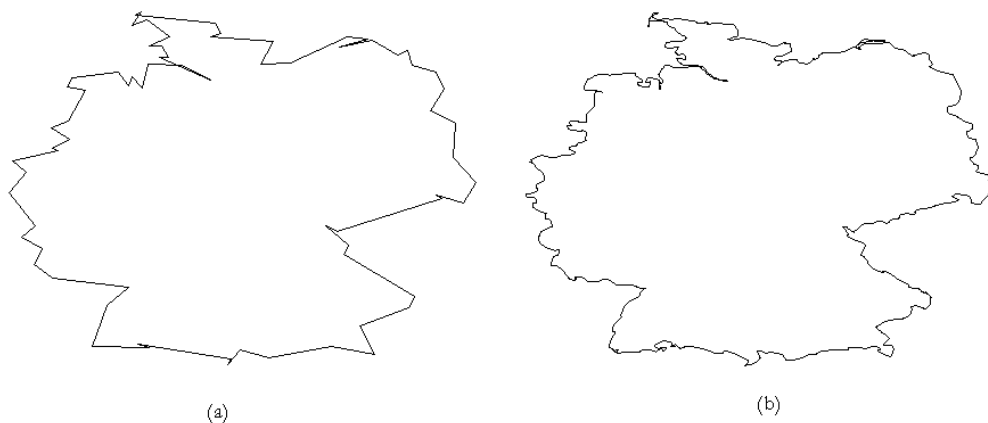


Figure 4.3 Two shapes at different levels of details

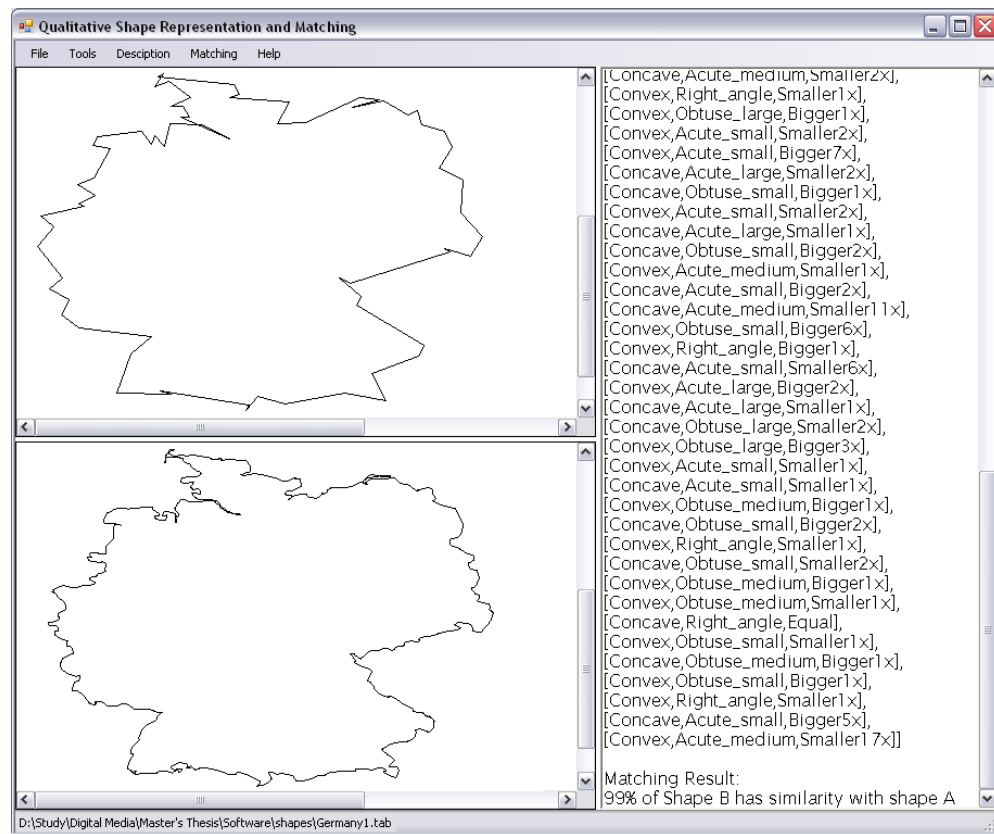


Figure 4.4 Matching results of two shapes at different levels of details

CHAPTER 5

5. IMPLEMENTATION

This chapter presents the software application developed for this thesis. Then it describes implementation procedures for construction of qualitative representation from a vector shape.

5.1 SOFTWARE APPLICATION

A software application is developed to implement the theory discussed in this thesis for academic purpose. This is a simple application to transform the shape from its vector representation to its qualitative representation. And then match a given shape with a database contains qualitative representations of shapes. This application also provides a drawing tool to construct shapes and saves the constructed shapes as vector files. Figure 5.1 shows a snapshot of the application.

This software application is developed under Microsoft Dot Net platform in C# language. A copy of this software is available in the CD attached with this thesis. Please find a user manual for the software at the end of this document in the appendix B.

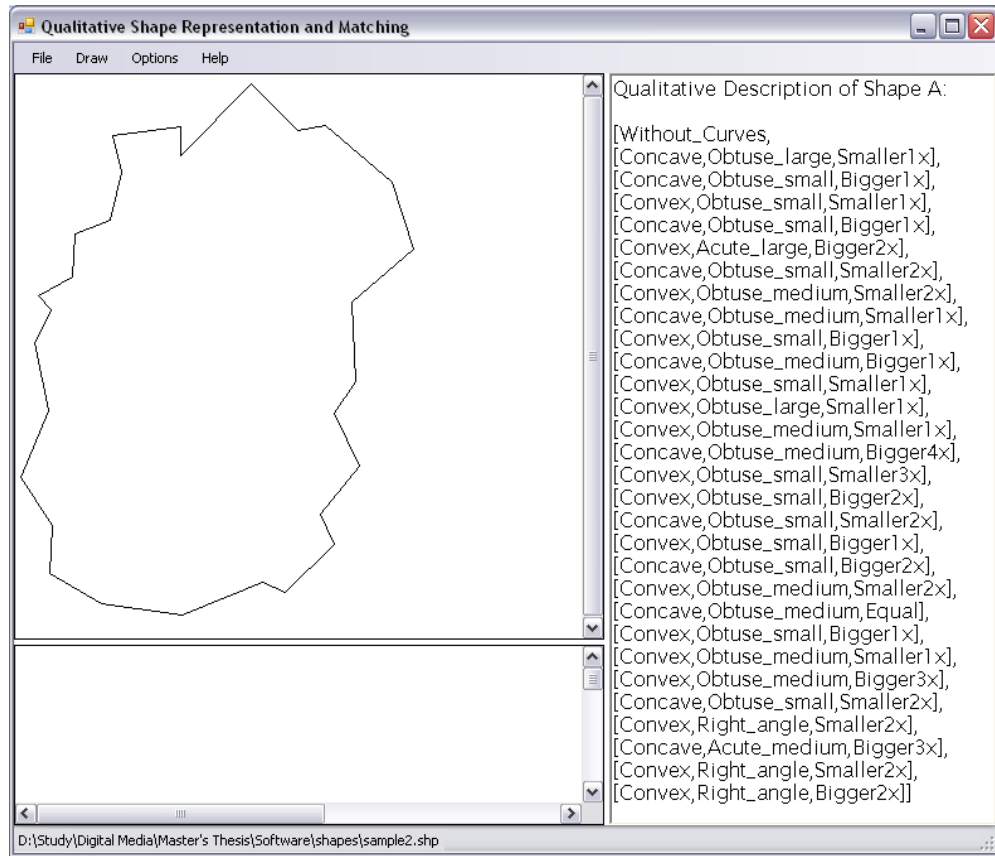


Figure 5.1 Snapshot of software application

This software application do not take image files as input because image files involve much of image processing procedures to convert an image shape in a vector form therefore it goes out of the scope of this thesis. However, this application takes vector files as input which contains all the vertices of a given shape. Our software application provides a shape construction tool which is very user friendly and easy to use to construct shapes on certain requirements and then saves them in files as vector form. Figure 5.2 shows a snap shot of shape drawing tool.

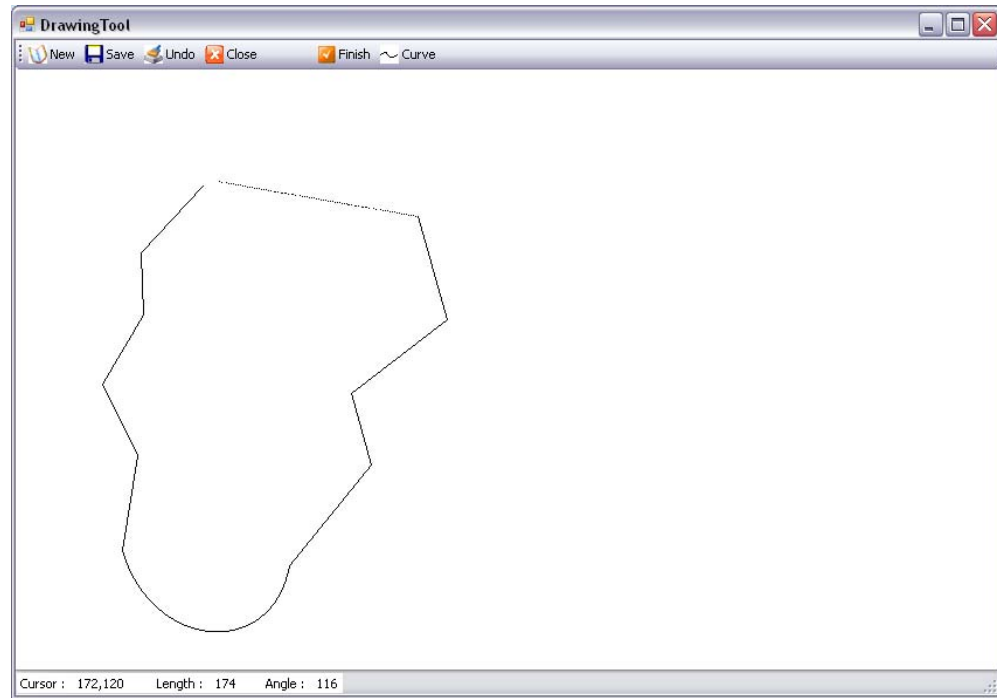


Figure 5.2 Snapshot of shape drawing tool

5.2 CONSTRUCTION OF QUALITATIVE DESCRIPTION

We construct qualitative description from vertices of shapes. For a given shape in its vector form to construct its qualitative description we apply different procedures. First, the vertices of input shape are loaded in an array. Then each vertex is evaluated for its qualitative description. Shapes that contain curves are more complex than straight vectors. First we explain the procedure for constructing qualitative description of curve segments.

5.2.1 CONSTRUCTION OF CURVE'S DESCRIPTION

As described in chapter 3, we illustrate four features of a curve in its description i.e. [*curve*, *convexity*, *curvature type*, *relative length*]. Symbol '*curve*' is used to distinguish

curves from other vertices in the description of a shape. Therefore, the vertex (-1,-1) in the input file indicates the start of a curve and next four points belongs to this curve. We use quadratic Bezier curves to represent the curves in this thesis. Quadratic Bezier curve consists of four points i.e. starting point, end point and two control points. For constructing the description of a curve we need three relevant points they are first and last point of the curve and the point of maximum curvature. But the point of maximum curvature is not given; therefore first step would be to find the point of maximum curvature, to do this procedure is as follows:

1. First we find the general equation of the straight line passing through the first and last point of the curve in the form of $Ax + By = C$. Where A, B and C are the constants and are obtained by the following equations:

$$A = y_{p2} - y_{p1}$$

$$B = x_{p1} - x_{p2}$$

$$C = A x_{p1} + B y_{p1}$$

Where, p1 and p2 are the first and last point of the curve

2. Then, we calculate the perpendicular distance between each point of the curve to the straight line passing through first and the last point of the curve. The points on the curve are calculated using quadratic Bezier equation (i) given below and perpendicular distances from the straight line are calculated using equation (ii) given below:

$$(i) \quad x = (1 - u)^3 x_{p1} + 3u(1 - u)^2 x_{c1} + 3u^2(1 - u) x_{c2} + u^3 x_{p2}$$

$$y = (1 - u)^3 y_{p1} + 3u(1 - u)^2 y_{c1} + 3u^2(1 - u) y_{c2} + u^3 y_{p2}$$

where, u varies from 0.01 to 1.0

(x_{p1}, y_{p1}) and (x_{p2}, y_{p2}) are the first and last points of the curve

(x_{u1}, y_{u1}) and (x_{u2}, y_{u2}) are two control points of the curve

$$(ii) \quad Dist = \left(\frac{Am+Bn+C}{\sqrt{A^2+B^2}} \right)$$

where, (m, n) is the point on the curve

and A, B, C are the constants of oriented line equation, $Ax + By + C = 0$

After calculating the point of maximum curvature, we have all three points of relevance to construct the description of a curve. For any curve the description is constructed against the point of maximum curvature that is also called reference points of the curve. Therefore next step is to find the convexity of the given curve through following procedure:

- Convexity of a reference point is determined using the equation given below that calculates the position of a point (reference point) with respect to straight line passing through the first and last point of the curve. Convexity can be either concave or convex with respect to the position of the reference point. If reference point lies on left side of the oriented line then convexity of reference point is convex. On the other hand if reference point lies on the right side of the oriented line then convexity of reference point would be concave.

$$f(x, y) = (x - x_{p1})(y_{p2} - y_{p1}) - (y - y_{p1})(x_{p2} - x_{p1})$$

when $f(x, y) > 0 \rightarrow \text{concave}$

when $f(x, y) < 0 \rightarrow \text{convex}$

where, (x, y) is reference point and $(x_{p1}, y_{p1}), (x_{p2}, y_{p2})$ are the first and last points of the curve.

Next qualitative feature of a curve for its description is curvature type. For describing curvature type we first find a point P_k , midpoint of the oriented line from initial point to final point of the curve. Then we compare two distances D_a , distance from initial point of the curve to the point P_k , and D_b , distance from point P_k to maximum curvature point. Point is calculated using following equation:

$$Pk(x, y) = \left(\frac{x_{p1} + x_{p2}}{2}, \frac{y_{p1} + y_{p2}}{2} \right)$$

D_a and D_b are compared to find the type of curvature of a curve. When D_a is equal to D_b then curvature type is semicircle. If D_a is less than D_b then Curvature type is acute. Similarly relative length is computed by comparing these two distances for how many times D_a is bigger or smaller than D_b . This feature seems very quantitative but it is not because, for example two times does not mean exactly two times however, it means D_a is greater than one and half times of D_b and less than two and half times of D_b .

5.2.2 CONSTRUCTION OF NON-CURVE VERTEX'S DESCRIPTION

Description of non curve vertex consists of three qualitative features; they are convexity, angle and relative length of two connected edges. For describing a vertex, called reference vertex, two other relevant vertices, previous vertex and following vertex, are required. First step is to determine convexity of the reference vertex:

- Convexity of a reference vertex is determined by drawing an oriented line from previous vertex called i , to following vertex called j . Then position of reference vertex is calculated with respect to the oriented line using formula given below. If reference vertex remains on the left side of the oriented line then the convexity is concave, on the other hand if it remains on right side of the oriented line then the convexity is convex.

$$f(x, y) = (x - x_i)(y_j - y_i) - (y - y_i)(x_j - x_i)$$

when $f(x, y) > 0 \rightarrow \text{concave}$

when $f(x, y) < 0 \rightarrow \text{convex}$

- Angle of a reference vertex is determined using non-right triangle trigonometry. A triangle is constructed from vertex i , vertex j and vertex k , as shown in figure 5.3. Lengths of all three sides are calculated using Euclidean distance formula. Then angle of reference vertex j is determined using cosine equation given below:

$$\cos(j) = \frac{I^2 + K^2 - J^2}{2IK}$$

$$\text{OR, } j = \cos^{-1} \left(\frac{I^2 + K^2 - J^2}{2IK} \right)$$

where, I, J and K are three sides of triangle

Then angle j is compared against seven qualitative groups of angles [acute-small, acute-medium, acute-large, right-angle, obtuse-small, obtuse-medium, obtuse-large] described in chapter 3 and determined accordingly.

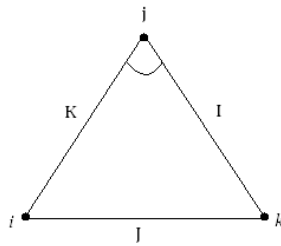


Figure 5.3 Triangle IJK formed by three vertices i, j and k

- Relative length of two adjacent edges of reference vertex is calculated by comparing their lengths. Description of relative length contains two types of information, first is to determine if the edge is smaller, bigger or equal than the following edge, and second information determines the quality of how much bigger or smaller. Lengths of edges are simply calculated using Euclidean distance formula. If first edge is greater than or less than by ten percent of its half length from second edge then qualitative relative length is equal. Similarly, if first length is smaller than by fifty percent of its length from the length of second edge then relative length is smaller2x.

```
IF LengthA + 10%LengthA < LengthB & LengthA - 10%LengthA > LengthB
    THEN LengthA, LengthB are Equal
END IF
ELSE IF LengthA < LengthB THEN
```

```

NoOfTimes =  $\frac{\text{LengthB}}{\text{LengthA}}$ 
Remainder = LengthB % LengthA
IF Remainder ≤ 50%LengthA THEN
    RelativeLength = NoOfTimes Smaller
ELSE
    RelativeLength = (NoOfTimes + 1) Smaller
END IF
END IF
ELSE IF LengthA > LengthB THEN
    NoOfTimes =  $\frac{\text{LengthA}}{\text{LengthB}}$ 
    Remainder = LengthA % LengthB
    IF Remainder ≤ 50%LengthB THEN
        RelativeLength = NoOfTimes Bigger
    ELSE
        RelativeLength = (NoOfTimes + 1) Bigger
    END IF
END IF
END IF

```

CHAPTER 6

6. CONCLUSION AND FUTURE WORK

In this last chapter we conclude our work and present the main contribution of our work. At the end we highlight some paths for future research.

6.1 CONCLUSION

Qualitative representation of shapes is an important concern in many areas including geographic information systems, robotic navigation, high level visual processing , engineering design, common sense reasoning about physical systems and specifying visual language and semantics. However, in this thesis we are interested in shapes related to geographic information systems. And GIS involve large amount of shapes data where matching of those shape related data is one of the major tasks that are required to perform frequently. Moreover spatial input may exist at different level of detail than the information stored in geographic database or spatial input may not be complete.

The main contribution of this thesis is that it provides a cognitively plausible extension of an existing qualitative representation scheme that precisely describes qualitative features of shapes to support complicated representation and matching tasks. This extended qualitative theory of shape description is more cognitive in a sense that it can distinguish cognitively distinguishable shapes which was not always possible with most of the existing approaches. This qualitative theory supports wide range of shapes including regular, non-regular polygons and shapes including curvilinear segments.

Main interest of this thesis was to implement a qualitative theory to represent geographical shapes like boundaries of countries, cities, lakes etc. Boundaries found in different geographical systems are not same and even exist at different granularity level. And reconstruction of boundaries from digital images often causes displacement of pixels due to digitization effect and noise in the images. Sometimes shapes encountered in GIS may not be complete or may be obscured by some other shapes. Therefore, quantitative models are not suitable for description and matching of such shapes. We have provided a qualitative shape theory to deal with such complicated situations.

We have developed an application for representation and matching of shapes where we have successfully applied our qualitative shape theory on geographical boundaries. Application has a database of qualitative representations of shapes and user can select a shape to find its best match from the database. Application returns the best match with its matching results. The input shape which is given to the system for matching can be at finer level of granularity or may not be complete but the application will adjust its granularity level and will apply partial matching algorithm to get best results.

6.2 FUTURE WORK

Here we highlight some tasks and directions for further work in this qualitative shape theory.

Shape representation theory can be further extended to incorporate more curvilinear shapes. Currently it uses quadratic Bezier curves to represent all types of curves therefore it contains some constraints of representing complex curves. The procedure of constructing qualitative description of curves can also be extended because currently it follows the same principal of three relevant points used for non curvilinear segments and constructs the description complete curve for maximum curvature point.

A qualitative algorithm can be developed for coarsening qualitative representation of shapes like DCE does for vector shapes. The shapes stored in database are in the form of their qualitative representation therefore they cannot be transformed to further coarser level by applying DCE algorithm because DCE algorithm only works for vector representation of shapes. Consequently a qualitative algorithm for coarsening qualitative representations of shapes is required. Qualitative coarsening algorithm can be very useful when to compare a shape which is coarser then the shape in database.

Moreover, matching algorithm can be extended for different similarity levels for example while matching two angles which are not same but are neighbors can be given a similarity value. For example while comparing two vertices of two shapes, if they are convex and share the same relative length description but qualitative angles of both vertices are not same however they are neighboring angles. In this case both vertices have a cognitive similarity which can be taken into account in shape matching algorithm. Secondly; matching algorithm has to be optimized to support very large shapes.

BIBLIOGRAPHY

[Ballard and Brown 1982]

Ballard, D. H. & Brown, C. M. (1982), *Computer Vision*, Prentice-Hall, Inc.

[Biederman 1987]

Biederman, I. (1987), 'Recognition-by-components: A theory of human image understanding', *Psychological Review* 94(2), 115.147.

[Boyle and Thomas 1988]

Boyle, R. D. & Thomas, R. C. (1988), *Computer Vision: A First Course*, Blackwell Scientific Publications.

[Cohn 1997]

Cohn, A. G. (1997), Qualitative spatial representation and reasoning techniques, in G. Brewka, C. Habel & B. Nebel, eds, 'Proceedings of KI-97', Vol. 1303 of LNAI, Springer-Verlag, pp. 1.30.

[Cohn and Hazarika 2001]

Cohn, A. G & Hazarika, S .M. (2001) , 'Qualitative spatial representation and reasoning, An overview', *Fundamenta Informatica* 43 (2001) 2-32 IOS Press, pp. 21.

[Freska and Zimmermann 1992]

Freska, C. and Zimmermann, K. , "On the utilization of spatial structures for cognitively plausible and efficient reasoning", In *IEEE International Conference on Systems, Man and Cybernetics*, Chicago, 1992.

[Freksa and Röhrig 1993]

Freksa, C. & Röhrig, R. (1993), Dimensions of qualitative spatial reasoning, in N. P. Carret 'e & M. G. Singh, eds, 'Qualitative Reasoning and Decision Technologies', CIMNE, pp. 483.492.

[Galton and Meathrel 1999]

Galton, A and Meathrel, R. C., "Qualitative outline theory", In *IJCAI-99*, pages 1061–1066, Stockholm, Sweden, 1999.

[Galton and Meathrel 2000]

Galton, A. and Meathrel, R. C. ," Qualitative representation of planar outlines", In *Proc. Of 14th ECAI*, pages 224–228. IOS Press, 2000.

[Gottfried 2002]

Gottfried, B., "Tripartite Line Tracks", In K. Wojciechowski, editor, *International Conference on Computer Vision and Graphics*, pages 288–293, Zakopane, 2002.

[Gottfried 2005]

Gottfried, B., "Global Feature Schemes for Qualitative Shape Descriptions," in *Proc. IJCAI 05 WS Spatial-Temporal Reasoning*, L. G. H. W. Guesgen C. Freksa, Ed., 2005.

[Hernandez 1994]

Hernandez, D. (1994), Qualitative Representation of Spatial Knowledge, Vol. 804 of Lecture Notes in Artificial Intelligence, Springer-Verlag.

[Hobbs 1985]

Hobbs, J.R. 1985. Granularity. In A. Joshi (Ed.), Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Los Angeles, CA, 432–435.

[Hoffman and Richards 1982]

Hoffman, D. D. & Richards, W. A. (1982), Representing smooth plane curves for recognition: implications for figure-ground reversal, in 'Proceedings of AAAI-82', American Association for Artificial Intelligence, pp. 5.8.

[Hoffman and Richards 1984]

Hoffman, D. D. & Richards, W. A. (1984), 'Parts of recognition', *Cognition* 18, 65.96.

[Jungert, 1993]

E. Jungert. Symbolic spatial reasoning on object shapes for qualitative matching. In *COSIT 1993*, pages 444–462, Island of Elba, 1993. Springer.

[Latecki and Lakaemper 1999]

Latecki, L.J. and Lakaemper, R., Polygon Evolution by Vertex Deletion, Scale-Space Theories in Computer Vision Proc. Int'l Conf. Scale-Space '99, M. Nielsen, P. Johansen, O.F. Olsen, and J. Weickert, eds., Sept. 1999.

[Layton 1988]

Leyton, M. (1988), 'A process-grammar for shape', *Artificial Intelligence* 34, 213.247.

[Museros and Escrig 2004]

Museros, Lledó Cabedo and Escrig, M. Teresa, A Qualitative Theory for Shape Representation and Matching for Design. ECAI 2004: 858-862

[Marr and Nishihara 1978]

Marr, D. & Nishihara, H. K. (1978), 'Representation and recognition of the spatial organization of three-dimensional shapes', Proceedings of the Royal Society of London, Series B 200, 269.294

[Pizer et al. 1987]

Pizer, S. M., Oliver, W. R. & Bloomberg, S. H. (1987), 'Hierarchical shape description via the multiresolution symmetric axis transform', IEEE Transactions on Pattern Analysis and Machine Intelligence 9(4), 505.511.

[Schuldt et al. 2006]

A. Schuldt, B. Gottfried, and O. Herzog, "A Compact Shape Representation for Linear Geographical Objects: The Scope Histogram," in *Proc. 14th ACM International Symposium on Advances in Geographic Information Systems*, S. Nittel and R. A. de By, Eds., Arlington, VA, USA, Nov. 10-11, 2006, pp. 51–58.

[Shokoufandeh et al. 2002]

A. Shokoufandeh, S. Dickinson, C. Jonsson, L. Bretzner, and T. Lindeberg. On the Representation and Matching of Qualitative Shape at Multiple Scales. In Proceedings, European Conference on Computer Vision, Copenhagen, May 2002.

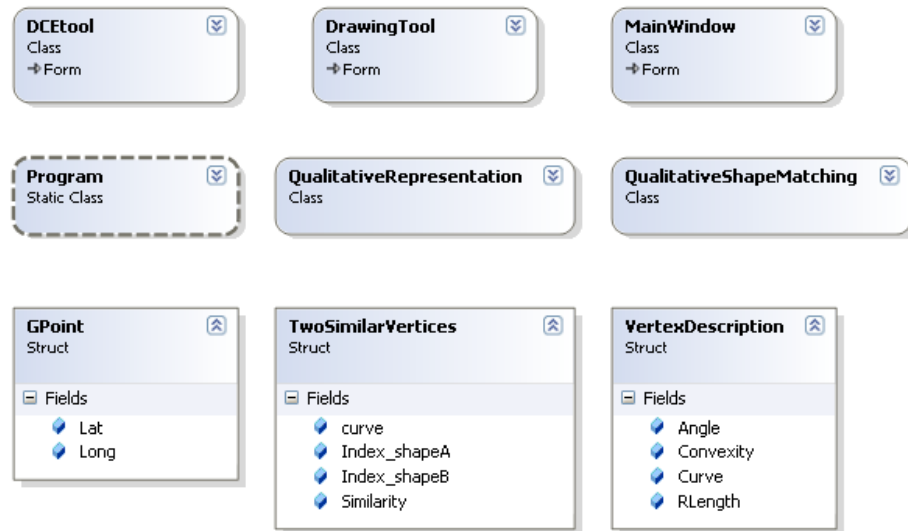
[Veltkamp and Hagedoorn 1999]

Veltkamp, R. C. and Hagedoorn, M., State of the art in shape matching.
Technical Report UU-CS-1999-27, Utrecht, 1999.

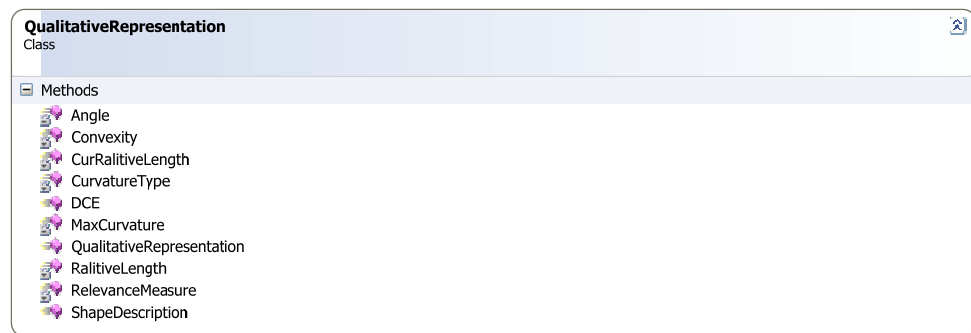
APPENDIX A

SOURCE CODE FOR QUALITATIVE REPRESENTATION AND MATCHING ALGORITHMS

Following diagram is a class diagram of application provided with this thesis. Where static class program is main class which starts the application and class MainWindow is for main window appears when we run the application. In the third row of figure there are three structures which we have used in this application. Two classes for qualitative shape representation and matching are class QualitativeRepresentation and class QualitativeShapeMatching.



Source code: class QualitativeRepresentation



```

class QualitativeRepresentation
{
    public QualitativeRepresentation()
    {
    }
    public String ShapeDescription(ArrayList Vertices)
    {
        Point VertexI, VertexJ, VertexK;
        String DescriptionStr=" [Without_Curves, ";
        Vertices.RemoveAt(Vertices.Count - 1);
        bool curve = false;

        for (int I = 0; I < Vertices.Count;i++ )
        {
            String VertexDescription="";
            if (curve == true)
            {
                DescriptionStr = DescriptionStr.Replace("Without_Curves",
                    "With_Curves");
                if (I + 4 < Vertices.Count - 1)
                {
                    VertexJ = MaxCurvature((Point)Vertices[I + 1],
                        (Point)Vertices[I + 2], (Point)Vertices[I + 3],
                        (Point)Vertices[I + 4]);

                    VertexI = (Point)Vertices[I + 1];
                    VertexK = (Point)Vertices[I + 4];
                    VertexDescription = "[" + "Curve, ";
                    VertexDescription = VertexDescription +

```

```

        Convexity(VertexI, VertexJ,
        VertexK) + ",";
VertexDescription = VertexDescription +
        CurvatureType(VertexI, VertexJ,
        VertexK) + ",";
VertexDescription = VertexDescription +
        CurRalitiveLength(VertexI, VertexJ, VertexK) + "];";
I = I + 4;
curve = false;
}
// when curve is at the end of the vertices array

else if (I + 3 == Vertices.Count - 1)
{
    VertexJ = MaxCurvature((Point)Vertices[I + 1],
        (Point)Vertices[I + 2], (Point)Vertices[I +
        3], (Point)Vertices[0]);

    VertexI = (Point)Vertices[I + 1];
    VertexK = (Point)Vertices[0];
    VertexDescription = "[" + "Curve,";
    VertexDescription = VertexDescription +
        Convexity(VertexI, VertexJ,
        VertexK) + ",";
    VertexDescription = VertexDescription +
        CurvatureType(VertexI, VertexJ,
        VertexK) + ",";
    VertexDescription = VertexDescription +
        CurRalitiveLength(VertexI,
        VertexJ, VertexK) + "];";

    I = I + 3;
    curve = false;
}
}
else
{
    VertexJ = (Point)Vertices[i];
    if (I < Vertices.Count - 1)// if I is not last vertex
    {
        VertexK = (Point)Vertices[I + 1];
        // Check vertex k if it is start of a curve
        if (VertexK.X == -1 && VertexK.Y == -1)
        {
            VertexK = (Point)Vertices[I + 2];
            curve = true;
        }
    }
    else// if I is last vertex then k would be first vertices
        VertexK = (Point)Vertices[0];

    if (I > 0)
        VertexI = (Point)Vertices[I - 1];
    // When first vertex is reference then vertexI would be the last one
    else
    {
        //check if last reference of the array is array
        if (Vertices.Count - 4 > 0)
        {
            Point pt = (Point)Vertices[Vertices.Count - 4];

```

```

        if (pt.X == -1 && pt.Y == -1)
        {
            continue;
        }
    }
    VertexI = (Point)Vertices[Vertices.Count - 1];
}

VertexDescription = "[" + Convexity(VertexI, VertexJ,
    VertexK) + "," + Angle(VertexI, VertexJ, VertexK) +
    "," + RalitiveLength(VertexI, VertexJ, VertexK) +
    "];
}
if (I == Vertices.Count - 1)
    DescriptionStr = DescriptionStr + VertexDescription ;
else
    DescriptionStr = DescriptionStr + VertexDescription +
        ",";
}
DescriptionStr += "];
Vertices.Add( Vertices[0]);
return DescriptionStr;
}

public void DCE(ArrayList verticesIn)
{
    Point VertexI, VertexJ, VertexK;
    verticesIn.RemoveAt(verticesIn.Count - 1);
    int[] K = new int[verticesIn.Count];
    bool curve = false;

    for (int I = 0; I < verticesIn.Count; i++)
    {
        if (curve == true)
        {
            if (I + 4 < verticesIn.Count - 1)
            {
                K[i] = -1; K[I + 1] = -1; K[I + 2] = -1; K[I + 3]=-1;
                K[I + 4] = -1;
                I = I + 4;
                curve = false;
            }
            else if (I + 3 == verticesIn.Count - 1)
            // when curve is at the end of the vertices array
            {
                K[i] = -1; K[I + 1] = -1; K[I + 2] = -1;
                K[I + 3] = -1;

                I = I + 3;
                curve = false;
            }
        }
        else
        {
            VertexJ = (Point)verticesIn[i];
            if (I < verticesIn.Count - 1)// if I is not last vertex
            {
                VertexK = (Point)verticesIn[I + 1];
                if (VertexK.X == -1 && VertexK.Y == -1)

```

```

        // Check vertex k if it is start of a curve
        {
            VertexK = (Point)verticesIn[I + 2];
            curve = true;
        }
    }
    else// if I is last vertex then k would be first vertices
        VertexK = (Point)verticesIn[0];

    if (I > 0)
        VertexI = (Point)verticesIn[I - 1];
    else
        // When first vertex is reference then vertexI would be the
        last one
        {
            //check if last reference of the array is curve
            if (verticesIn.Count - 4 > 0)
            {
                Point pt=(Point)verticesIn[verticesIn.Count - 4];
                if (pt.X == -1 && pt.Y == -1)
                {
                    K[i] = -1;
                    continue;
                }
            }
            VertexI = (Point)verticesIn[verticesIn.Count - 1];
        }

    K[i] = RelevanceMeasure(VertexI, VertexJ, VertexK);
}
}
verticesIn.Add(verticesIn[0]);
int j = 0;
int Least,index=0;
while (K[j] == -1) {j++;}
Least = K[j];
for ( j = 0; j < verticesIn.Count-1; j++)
{
    if (K[j] < Least)
    {
        Least = K[j];
        index = j;
    }
}
verticesIn.RemoveAt(index);
// return verticesIn;
}
private int RelevanceMeasure(Point Vi, Point Vj, Point Vk)
{
    int LenA = (int)Math.Sqrt(Math.Pow((Vk.X - Vj.X), 2) +
        Math.Pow((Vk.Y - Vj.Y), 2));
    int LenB = (int)Math.Sqrt(Math.Pow((Vi.X - Vj.X), 2) +
        Math.Pow((Vi.Y - Vj.Y), 2));
    int LenC = (int)Math.Sqrt(Math.Pow((Vi.X - Vk.X), 2) +
        Math.Pow((Vi.Y - Vk.Y), 2));
    double angle;
    int iAngle;
    int denominator = (2 * LenA * LenB);
    if (denominator > 0)
        angle = Math.Acos((Math.Pow(LenA, 2) + Math.Pow(LenB, 2) -
            Math.Pow(LenC, 2)) / denominator);
    else

```

```

        angle = Math.Acos((Math.Pow(LenA, 2) + Math.Pow(LenB, 2) -
            Math.Pow(LenC, 2)));

        iAngle = (int)(angle * 180 / Math.PI);
        iAngle = 180 - iAngle;

        int k = (int)((iAngle * LenA * LenB) / (LenA + LenB));
        return k;
    }
    private String Convexity(Point Vi, Point Vj, Point Vk)
    {
        int convexity = (Vj.X - Vi.X) * (Vk.Y - Vi.Y) - (Vj.Y - Vi.Y) *
            (Vk.X - Vi.X);
        if (convexity > 0)
        {
            return "Concave";
        }
        else if (convexity < 0)
        {
            return "Convex";
        }
        else
            return "OnLine";
    }
    private String Angle(Point Vi, Point Vj, Point Vk)
    {
        //this code calculates the angle
        int LenA = (int)Math.Sqrt(Math.Pow((Vk.X - Vj.X), 2) +
            Math.Pow((Vk.Y - Vj.Y), 2));
        int LenB = (int)Math.Sqrt(Math.Pow((Vi.X - Vj.X), 2) +
            Math.Pow((Vi.Y - Vj.Y), 2));
        int LenC = (int)Math.Sqrt(Math.Pow((Vi.X - Vk.X), 2) +
            Math.Pow((Vi.Y - Vk.Y), 2));
        double angle;
        int iAngle;
        int denominator = (2 * LenA * LenB);
        if (denominator > 0)
            angle = Math.Acos((Math.Pow(LenA, 2) + Math.Pow(LenB, 2) -
                Math.Pow(LenC, 2)) / denominator);
        else
            angle = Math.Acos((Math.Pow(LenA, 2) + Math.Pow(LenB, 2) -
                Math.Pow(LenC, 2)));

        iAngle = (int)(angle * 180 / Math.PI);

        if (iAngle > 0 && iAngle < 35)
        {
            return "Acute_small";
        }
        else if (iAngle >= 35 && iAngle < 55)
        {
            return "Acute_medium";
        }
        else if (iAngle >= 55 && iAngle < 80)
        {
            return "Acute_large";
        }
        else if (iAngle >= 80 && iAngle < 100)
        {

```

```

        return "Right_angle";
    }
    else if (iAngle >= 100 && iAngle < 125)
    {
        return "Obtuse_small";
    }
    else if (iAngle > 125 && iAngle < 145)
    {
        return "Obtuse_medium";
    }
    else if (iAngle >= 145 && iAngle < 180)
    {
        return "Obtuse_large";
    }
    return "Straight_Line";
}
private String RalitiveLength(Point Vi, Point Vj, Point Vk)
{
    int LenA = (int)Math.Sqrt(Math.Pow((Vi.X - Vj.X), 2) +
        Math.Pow((Vi.Y - Vj.Y), 2));
    int LenB = (int)Math.Sqrt(Math.Pow((Vk.X - Vj.X), 2) +
        Math.Pow((Vk.Y - Vj.Y), 2));
    int LenC = (int)Math.Sqrt(Math.Pow((Vi.X - Vk.X), 2) +
        Math.Pow((Vi.Y - Vk.Y), 2));
    int Percent5 = (int) (LenA * 0.05);
    if (LenB > (LenA - Percent5) && LenB < (LenA + Percent5))
    {
        return "Equal";
    }
    else if (LenA < LenB)
    {
        int Percent50 = (int)(LenA * 0.50);
        int times = LenB / LenA;
        int rem = LenB % LenA;
        if (rem <= Percent50)
        {
            return "Smaller" + times + "x";
        }
        else if (rem > (Percent50 ))
        {
            return "Smaller" + (times + 1) + "x";
        }
    }
    else if (LenA > LenB)
    {
        int Percent50 = (int)(LenB * 0.50);
        int times = LenA / LenB;
        int rem = LenA % LenB;
        if (rem <= Percent50)
        {
            return "Bigger" + times + "x";
        }
        else if (rem > (Percent50 ))
        {
            return "Bigger" + (times + 1) + "x";
        }
    }
    return "";
}
private Point MaxCurvature(Point p1, Point c1, Point c2, Point p2)
{

```

```

Point MaxCurPt = new Point();

//constants of line from first point of the curve to last point
int A1 = p2.Y - p1.Y;
int B1 = p1.X - p2.X;
int C1 = A1 * p1.X + B1 * p1.Y;
C1 = -1 * C1;

double u = 0.01;
int MaxDist = 0, Dist = 0;

int x = 0, y = 0;
while (u < 1)
{
// calculate a point on the curve using quadratic bezier curve equation
x = (int)((Math.Pow((1 - u), 3) * p1.X) + (3 * u *
    Math.Pow((1 - u), 2) * c1.X) + (3 * Math.Pow(u, 2) * (1 -
    u) * c2.X) + (Math.Pow(u, 3) * p2.X));
y = (int)((Math.Pow((1 - u), 3) * p1.Y) + (3 * u *
    Math.Pow((1 - u), 2) * c1.Y) + (3 * Math.Pow(u, 2) * (1 -
    u) * c2.Y) + (Math.Pow(u, 3) * p2.Y));

// calculate the distance of x,y from oriented line
double denominator= Math.Sqrt(Math.Pow((A1), 2) +
    Math.Pow((B1), 2));
if (denominator > 0)
    Dist = (int)((A1 * x + B1 * y + C1) / denominator);
Dist = (int)Math.Abs(Dist);
// find the maximum distance
if (Dist > MaxDist)
{
    MaxDist = Dist;
    MaxCurPt.X = x;
    MaxCurPt.Y = y;
}
u = u + 0.05;
}
// return maximum curvature point of the curve
return MaxCurPt;
}

private String CurvatureType(Point Vi, Point Vj, Point Vk)
{
    Point Pk = new Point();
// mid point of line from initial point of the curve to final point
Pk.X = (Vi.X + Vk.X) / 2;
Pk.Y = (Vi.Y + Vk.Y) / 2;

int Da = (int)Math.Sqrt(Math.Pow((Vi.X - Pk.X), 2) +
    Math.Pow((Vi.Y - Pk.Y), 2));
int Db = (int)Math.Sqrt(Math.Pow((Vj.X - Pk.X), 2) +
    Math.Pow((Vj.Y - Pk.Y), 2));

if (Da == Db)
    return "Semicircle";
else if (Da < Db)
    return "Acute";
else
    return "Plane";
}

```

```

private String CurRalitiveLength(Point Vi, Point Vj, Point Vk)
{
    Point Pk = new Point(); ;
    // mid point of line from initial point of the curve to final point
    Pk.X = (Vi.X + Vk.X) / 2;
    Pk.Y = (Vi.Y + Vk.Y) / 2;
    int LenA = (int)Math.Sqrt(Math.Pow((Vi.X - Pk.X), 2) +
        Math.Pow((Vi.Y - Pk.Y), 2));
    int LenB = (int)Math.Sqrt(Math.Pow((Vj.X - Pk.X), 2) +
        Math.Pow((Vj.Y - Pk.Y), 2));

    int Percent5 = (int)(LenA * 0.10);
    if (LenB > (LenA - Percent5) && LenB < (LenA + Percent5))
    {
        return "Equal";
    }
    else if (LenA < LenB)
    {
        int Percent50 = (int)(LenA * 0.50);
        int times = LenB / LenA;
        int rem = LenB % LenA;
        if (rem <= Percent50)
        {
            return "Smaller" + times + "x";
        }
        else if (rem > (Percent50))
        {
            return "Smaller" + (times + 1) + "x";
        }
    }
    else if (LenA > LenB)
    {
        int Percent50 = (int)(LenB * 0.50);
        int times = LenA / LenB;
        int rem = LenA % LenB;
        if (rem <= Percent50)
        {
            return "Bigger" + times + "x";
        }
        else if (rem > (Percent50))
        {
            return "Bigger" + (times + 1) + "x";
        }
    }
    return "";
}
}

class QualitativeShapeMatching
{
    public ArrayList AllMatched = new ArrayList();
    public int PartialCount = 0;
public QualitativeShapeMatching()
{
}

public ArrayList PartialMatching(String shapelStr, String shape2Str)
{
    ArrayList Qvertices_Shapel = new ArrayList();

```

```

ArrayList Qvertices_Shape2 = new ArrayList();
String Shape1Type = "", Shape2Type = "";

ArrayList PartialMatches = new ArrayList();
Qvertices_Shape1 = ExtractDescriptionFromString(shape1Str, ref
Shape1Type);
Qvertices_Shape2 = ExtractDescriptionFromString(shape2Str, ref
Shape2Type);
int m = Qvertices_Shape1.Count;// number of vertices of shape A
int n = Qvertices_Shape2.Count;// number of vertices of shape B
VertexDescription VertexA, VertexB; // container variables for
two vertices from two shapes for processing
TwoSimilarVertices Matched; // an object of structure to put
information of two matched vertices

// Array for finding largest common string
int[,] lcs = new int[m + 1, n + 1];
// array for consecutive vertices
int[,] w = new int[m + 1, n + 1];

int I, j, Similarity;

for (I = 1; I <= m; ++i)
{
    for (j = 1; j <= n; ++j)
    {
        VertexA = (VertexDescription)Qvertices_Shape1[I - 1];
        VertexB = (VertexDescription)Qvertices_Shape2[j - 1];
        Similarity = CompareTwoVertices(VertexA, VertexB);
        if (Similarity >= 70)
        {
            int k = w[I - 1, j - 1];

            lcs[I, j]=lcs[i-1, j- 1]+ConsecutiveMeasure(k + 1)-
                ConsecutiveMeasure(k);

            w[I, j] = k + 1;
        }
        else
        {
            lcs[I, j] = lcs[I - 1, j - 1];
        }

        if (lcs[I - 1, j] >= lcs[I, j])
        {
            lcs[I, j] = lcs[I - 1, j];

            w[I, j] = 0;
        }

        if (lcs[I, j - 1] >= lcs[I, j])
        {
            lcs[I, j] = lcs[I, j - 1];

            w[I, j] = 0;
        }
        if (w[I, j] > 0)
        {
            Matched.Index_shapeA = I - 1;
            Matched.Index_shapeB = j - 1;
            Matched.Similarity = Similarity;
        }
    }
}

```

```

        if (VertexA.Curve == VertexB.Curve && VertexB.Curve
            == "Curve")
            Matched.curve = true;
        else
            Matched.curve = false;
        AllMatched.Add(Matched);
    }
}
// perform backtracking

TwoSimilarVertices Match1,Match2; // an object of structure to
put information of two matched vertices
I = 0;
while( I < AllMatched.Count-1)
{
    bool pMatchflg = false;
    Match1 = (TwoSimilarVertices)AllMatched[i];
    j = I + 1;
    Match2 = (TwoSimilarVertices)AllMatched[j];
    while (Match2.Index_shapeA == Match1.Index_shapeA + 1 &&
        Match2.Index_shapeB == Match1.Index_shapeB + 1)
    {
        if (pMatchflg == false)
        {
            PartialMatches.Add(Match1);
            PartialCount++;
        }
        pMatchflg = true;
        PartialMatches.Add(Match2);
        Match1 = (TwoSimilarVertices)AllMatched[j++];
        if (j > AllMatched.Count - 1)
            break;
        Match2 = (TwoSimilarVertices)AllMatched[j];
    }
    I = j;
}

return PartialMatches;
}
}

```

APPENDIX B

1. USER MANUAL FOR THESIS SOFTWARE

Name of this software is ‘Qualitative Shape Representation and Matching’ available in the CD attached with thesis. This is a very simple application tool, developed for experimental purpose to implement theories presented in the thesis. Shapes data is given with the installation package. When installation is finished find Shapes data folder in your installed application folder. Shapes data folder contains three types of files i.e. *.tab, *.shp, *.qdb; where tab files contain geographic coordinates of boundaries of countries, .shp files contain simple x y coordinates of shapes and .qdb files are database files which contain the names of the files which are added in database. You can add more files in any database using main software application under Description-> Describe in Database menu.

2. SYSTEM REQUIREMENTS

- Windows XP or later
- Microsoft .Net framework

If your computer does not have .Net framework you need to install it before you run this application. .Net framework is freely available on internet you just need to download and install. Following is link of Microsoft website to download .Net framework. You can find further detail on the website.

<http://www.microsoft.com/downloads/details.aspx?FamilyID=10CC340B-F857-4A14-83F5-25634C3BF043&displaylang=en>

3. INSTALLATION INSTRUCTIONS

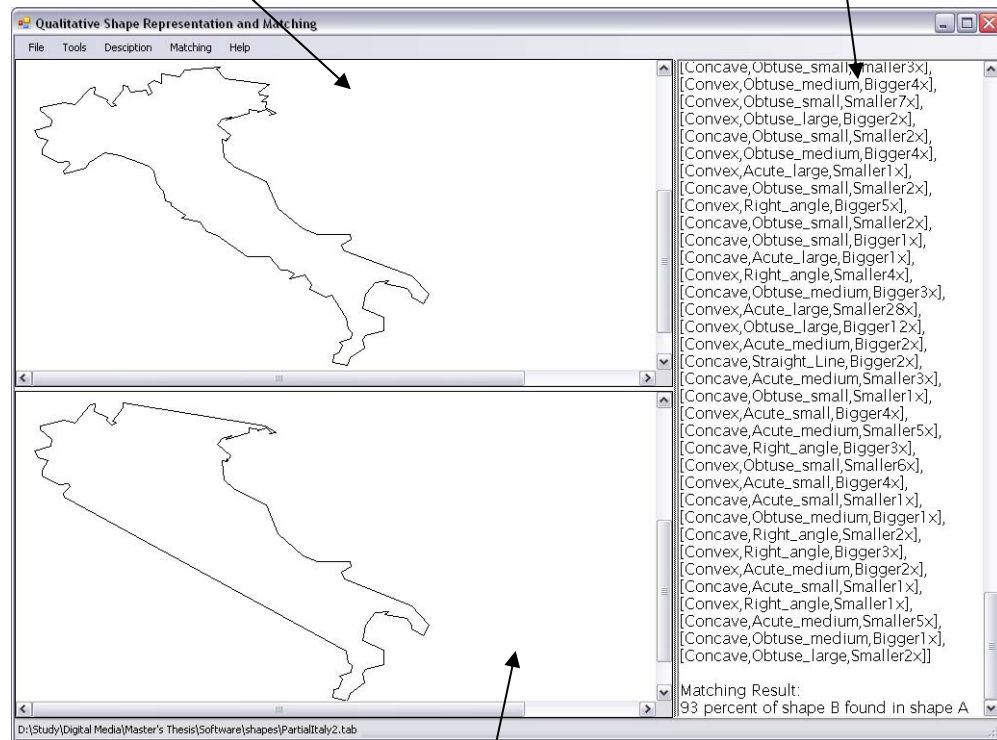
Insert CD provided with the thesis document in your CD ROM drive and follow the instruction below:

1. Open your CD drive
2. Open 'Qualitative shape representation and matching' folder
3. Click on Setup.exe file
4. Follow the instruction
5. Click close button when installation is finished
6. Run Qualitative Shape RM.exe to start application

4. MAIN WINDOW

Drawing Panel 1
It displays shapes which are opened

Text output display window



Drawing Panel 2
For drawing second shape

5. MENU OPTIONS

File

- Open and Describe

This menu option opens a shape and generates its qualitative. There are two types of files which can be opened in this application. First file type contains vector shapes and second file type contains geo-coordinates of countries maps.

- Open

This menu option opens a shape and draws it in the upper drawing panel. There are two types of files which can be opened in this application. First file type contains vector shapes and second file type contains geo-coordinates of countries maps.

- Exit

Tools

- Drawing Tool

This menu option opens drawing tool. Drawing can draw shapes and save them. For further details see under drawing tool.

- Coarsening Tool

This menu option opens coarsening tool. Coarsening tool is made for making shapes coarser using DCE algorithm. For further details see under coarsening tool.

Description

- Qualitative Representation

Qualitative Description menu option generates qualitative description of opened shape and displays it in text drawing panel.

- Describe and Store in Database

This menu option generates qualitative description of opened shape and stores it in database selected by user.

Matching

- Select and Match

Under this menu option user can select a shape for matching with already opened shape. Then qualitative description of selected shape and results of matching are displayed in the text panel.

- Select and Partial Match

Under this menu option user can select a shape for partial matching with already opened shape. Then qualitative description of selected shape and results of matching are displayed in the text panel.

- Match in Database

This menu option matches an opened shape in selected database and displays the results in text panel.

Help

- Help
- About

5. DRAWING TOOL

Drawing tool is a very simple application to draw different shapes. You just need to click in the drawing area of drawing tool window to start drawing. Then it provides you some useful drawing information, length of vector, angle with previous vector, to help you drawing your desired shape. Drawing tools also

supports curve drawing using Bezier curve, to draw a curve click on the curve button in the tool bar and click at the end point of the curve then set two control points of Bezier curve.

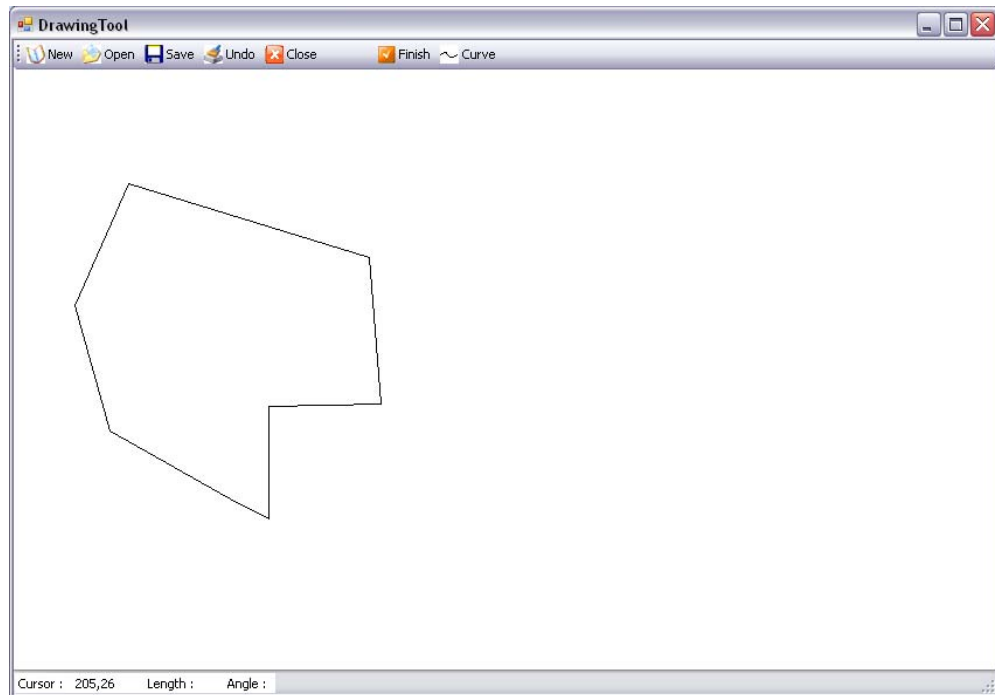


Figure 1 Snap shot of drawing tool

When you are finished with drawing your shape, click 'finish' button in the tool bar. Drawing tool only supports closed shapes therefore make sure your shape is closed before you press finish button. When you connect last vector of your shape with first vector, it is difficult to exactly click on the same point where first vector starts therefore this tool automatically connects your last click with the starting vertex of your shape if clicked nearby of first vertex in a radius of twenty pixels.

You can also open a shape in drawing tool and then redraw it with some changes in angles, lengths etc, see figure 2

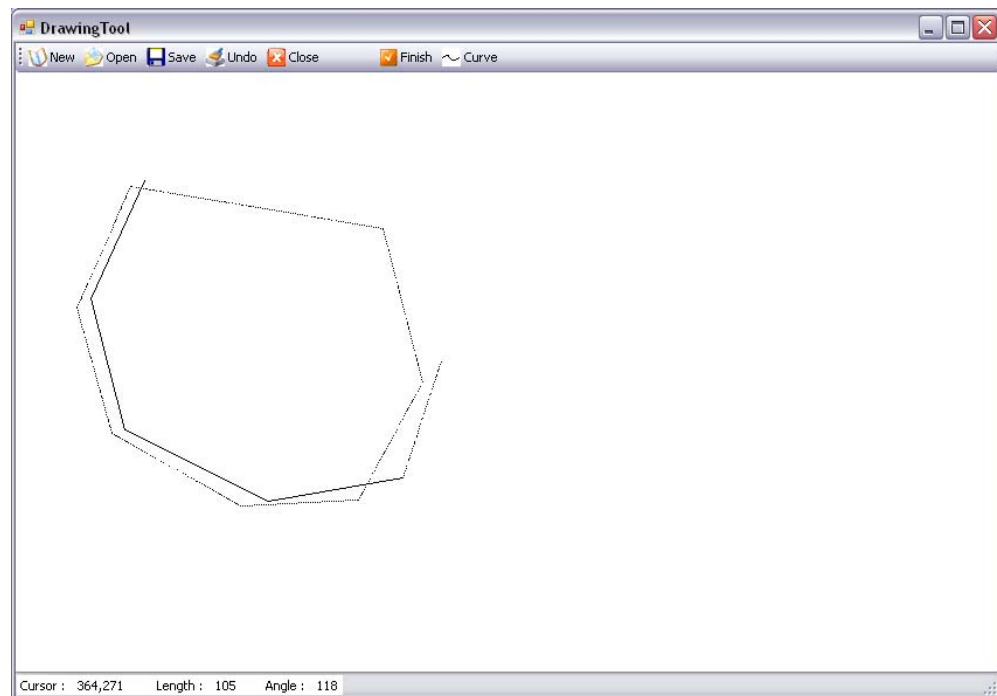


Figure 2 Snap shot of drawing tool

6. COARSENING TOOL

Coarsening tool is another very useful tool for this thesis to coarser given shapes at a given degree. This tool is very simple and easy to use. You just need to open a shape (a vector map in our case), enter a number in tools menu that how many vertices you want to remove to coarser the shape. This tool use DCE algorithm for shape coarsening. After entering the value, click 'Coarser' menu option in tools menu and the application will ask you for file name to save coarser version of shape.

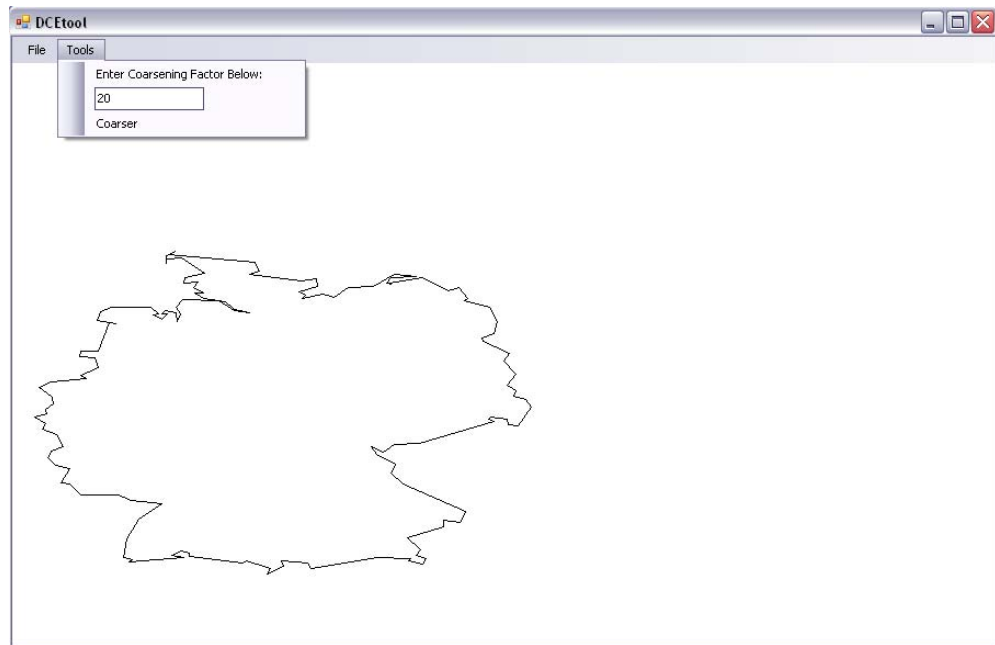


Figure 3 Snap shot of coarsening tool