Matching of Line Segments in 2-D Under Similarity Transformations

A dissertation submitted in partial fulfillment
of the requirements of M. Tech. (Computer Science)
degree of Indian Statistical Institute, Kolkata

by

Pradeep Kumar Chaturvedi

under the supervision of

Professor Bhargab B. Bhattacharya

Indian Statistical Institute
Kolkata-700108

June 2003

Indian Statistical Institute

205, Barrackpore Trunk Road,

Kolkata-700 108

Certificate of Approval

This is to certify that this thesis titled "**Matching of Line Segments in 2-D Under Similarity Transformations**" submitted by Pradeep Kumar Chaturvedi towards partial fulfillment of requirements for the degree of M. Tech in Computer Science at Indian Statistical Institute, Kolkata embodies the work done under my supervision.

B B Bhattacharya 8/7/03

Professor Bhargab B Bhattacharya
ACMU, Indian Statistical Institute,
Kolkata-700 108

Prof. B.B. Bhattacharya
Professor-in-Charge
Computer & Communication Sciences Division
Indian Statistical Institute
203, B.T. Road, Kolkata - 700 108

Acknowledgements

I take pleasure in expressing my deepest gratitude to Professor Bhargab B. Bhattacharya and Dr. Subhas Ch. Nandy for their guidance throughout the dissertation period. Their pleasant and encouraging words have always added to my spirits.

I am greatly indebted to Sri Arijit Bishnu for his careful and friendly attitude in guiding me through the difficult periods. My special thanks go to him.

I utilize this opportunity to thank my teachers and classmates.

Pradeep Kumar Chaturvedi,
Indian Statistical Institute, Kolkata
June, 2003.

Contents

Abstract

This report present a simple algorithm for the *matching of line segments in* 2D *under similarity transformations.* Given a set S of m line segments, called sample set, and a pattern set P of k line segments $(k \le m)$, the problem is to find a matching of P with a subset of S under similarity transformations. The proposed algorithm requires $O(n)$ space and preprocessing time $O(n\ log(n))$, and worst case query time complexity is $O(kn\log(n)+n^2)$. If set S consists of non parallel line segments then time complexity for such a case will reduce to $O(kn\log(n))$.

Chapter 1

Introduction

The matching and analysis of geometric patterns and shapes is of importance in various application areas, in particular in computer vision and pattern recognition, and also in other disciplines concerned with the form of objects such as cartography, molecular biology, and computer animation [6]. There are two general methods for image comparison: intensity-based (color and texture) and geometry-based (shape) [11].

The general situation for a matching problem is that we are given two objects A and B and we want to know how much they *resemble* each other. Usually one of the objects, say A, undergoes certain transformations like translation, rotation or scaling in order to be matched to B as close as possible. Important variants of this problem include partial matching i.e. when A resembles part of B, and a problem where for a given object A, the most similar one in a fixed set of objects (may be preprocessed) has to be found, e.g. in character or traffic sign recognition. There are other related problems e.g. simplification, shape interpolation, which come under the general class of 'Matching' problems [6]. Sometimes the complexity to solve such a problem exactly may be rather high, so that in such cases it makes sense to devise approximate algorithm which have acceptable complexity and gives the match within pre-specified accepted level of tolerance.

It is necessary to formally define the notions of objects, resemblance and transformations.

Formally 'Objects' are usually finite points ("point patterns") or "shapes" given in 2-D by polygons.

In order to measure "resemblance" various distance functions have been used, in particular much work has been done based on so-called *Hausdorff distance*[6].

What kind of geometric transformations are allowed to match objects A and B depends on the application. The simplest kind is certainly translations. The matching problem usually becomes much more difficult if we allow rotations and translations (these transformations are called rigid motions, or Euclidean transformations). Scaling means the linear transformation that "stretches" an object by a certain factor *'a'* about origin.

We call combinations of translations and scaling homothetics. A combination of euclidean transformations and scaling is called similarity transformations. The most general kind of transformations are arbitrary affine transformations which can occur e.g. in orthographic 2-dimentional projections of 3-dimensional objects [6].

There are various ways to approach the pattern-matching problem like Global image transforms, Image processing and Computational geometry. In this report we focus on methods from computational geometry. Computational geometry is the sub area of algorithm design that deals with the design and analysis of algorithms for geometric problems involving points, lines, polygons, and polyhedra etc. The standard approach taken in computational geometry is the development of exact, provably correct and efficient solutions to geometric problems [11].

Point Set Pattern Matching on a 2-D plane constitutes one of the most important matching problems. The present work "matching of Line Segments in 2-D under similarity transformations" is in some way a special case of this problem. Therefore first the Point Set pattern matching on 2-D is introduced followed by the problem that this work targets.

Point Set Pattern Matching on a 2-D plane

Given a sample set $P = \{p_1, p_2, \ldots\ldots, p_n\}$ consisting of n points, and a query set Q = $\{q_1, q_2, \ldots\ldots, q_n\}$ of k points in 2-D plane, where $k \leq n$, the objective is to find a k- subset M of P matching Q under similarity transformations provided such a match exists.

Rezende and Lee [9] have used computational geometry techniques to design an algorithm for finding a k-subset of P, that matches with Q, under similarity transformation. Their algorithm takes $O(n^2)$ space and total time complexity of partial point set pattern matching in 2-D is $O(kn^2)$.

Bishnu et al. [3] have proposed an improved algorithm for point set pattern matching under rigid motion in which reasonable preprocessing on data set can be performed so that the actual matching can be expedite significantly. The worst case time complexity for preprocessing is $O(n^2 \log(n))$ and space complexity is $O(n^2)$. Query processing in worst

case take total time of $O(kn^{4/3}\log(n))$. They have shown experimental results in their work, which shows improvement over Rezende and Lee algorithm.

Bishnu et al. [3] have tailored their algorithm for checking the congruence of a query set Q among the members in a sample set P where in both P and Q are line segments of non-zero length under rigid motion. The preprocessing time and space complexities are $O(n^2\log(n))$ and $O(n^2)$ respectively. Query processing in worst case takes $O(kn\log(n))$ time.

Matching of Line Segments in 2-D under similarity transformations

In this work we shall restrict our study to matching of line segments under similarity transformations in two dimensions only. Below, we formally define the domain of our problem.

Objects: Each of the objects A and B consists of a finite set of line segments in two dimensions.

Transformations: We are allowing translation, rotation and linear scaling and any combinations of these.

Match: Given a set A of k line segments and set B of m line segments $(k \leq m)$, A may be:

1. equal to a subset of B,
2. a translation of a subset of B,
3. a rotation of a subset of B,
4. a scaled copy of a subset of B.
5. equal to a transformed subset of B obtained by some combination of translation, rotation and scaling.

There are many variations of this problem:

Exact line segment pattern matching

Here we consider two sets A and B each having n line segments and A matches B exactly under similarity transformation.

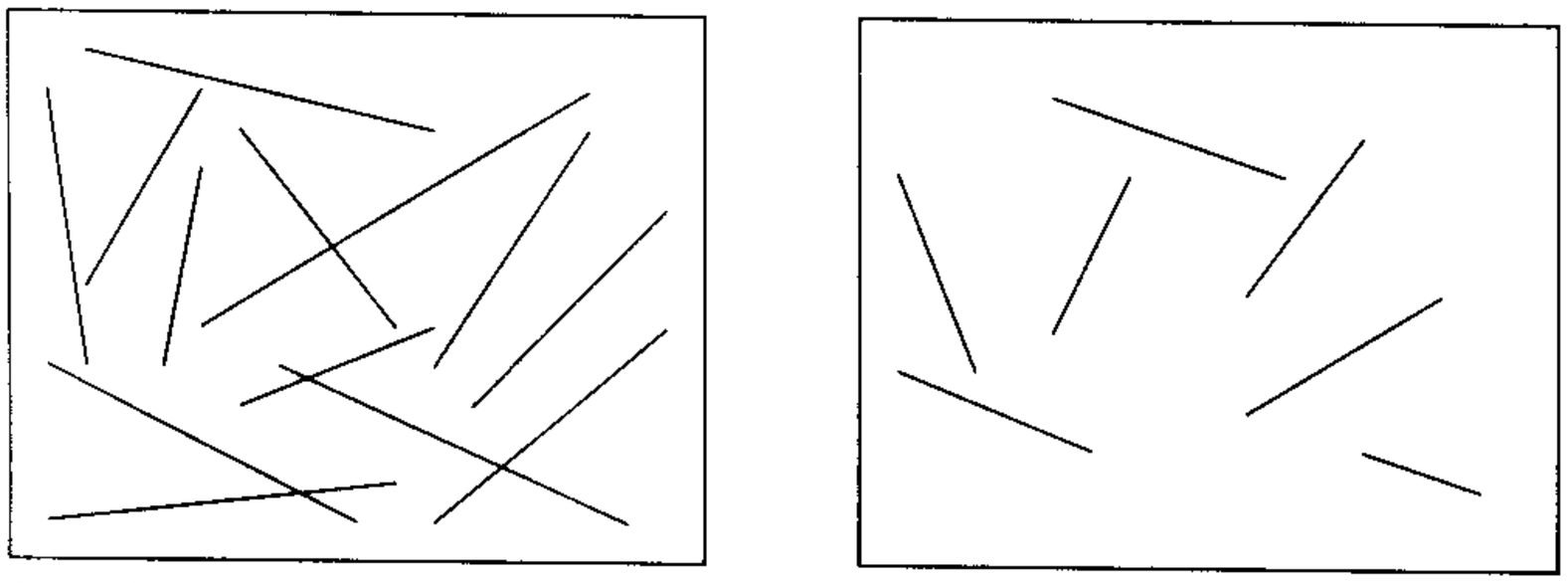
Approximate line segment pattern matching

The approximate line segment pattern matching is more realistic in actual application. Given two finite sets of line segments A and B of equal cardinality, the problem is to find an approximate matching under some transformation, i.e. for each line segment $b \in B$ find its match, $a \in A$ such that b lies in the ε-neighborhood of a (both end points of line segment b lies in a circle of radius ε centered at corresponding end point of line segment a). Variants of this problem are: (i) cardinality of A and B are equal and the objective is to check the existence of a one-to-one matching. (ii) more than one line segment of one set may be matched with the same line segment of the other set [7].

Subset matching

Let S be the larger set of line segments (called sample set) and P, a small set of line segments (called pattern set). The problem is to find whether P has a match in S, i.e. whether a subset of S exits which matches P under some transformation on P.

Present work: In this report, we present an algorithm in detail along with its complexity analysis for ***subset matching***, which also covers exact line segment pattern matching.



Sample set S

Pattern set P

Figure 1: Illustration of subset matching of line segments

Our objective is to know, in fig.1, whether there is a k-subset of sample set S that matches pattern set P under similarity transformations. The most trivial method is inspecting all possible $\binom{n}{k}$ subsets of P for a match with Q. This will lead to exponential time algorithm. In this report we are trying to build polynomial time algorithm.

Chapter 2

Proposed Algorithm and Data Structure

In this section, we propose an efficient algorithm for matching of line segments in two dimension under similarity transformations. Our algorithm supports subset matching, that is, the number of line segments in pattern set may be less than or equal to the number of line segment in sample set.

Our algorithm first preprocesses the sample set and then for a given arbitrary pattern set it tries to match the sample set with the given pattern set under some transformation. If a match exists the algorithm reports the correspondence between line segments in pattern set and line segment in sample set.

2.1 Preliminaries

A line segment in two dimensional plane can be represented by the coordinates of end points. But to facilitate comparison for matching, in addition to information about coordinates, we have incorporated additional fields pertaining to the slope and length of the line segment. We have restricted slope to lie in $[0,\pi)$. Each line is given an unique identification number.

The data structure for the line segment would contain the following field:

1. identification number
2. coordinates of the end points
3. slope
4. length

Let sample set $S = \{s_1, s_2, ..., s_n\}$, with |S| = n where s_i denotes data structure of i^{th} line segment in S and pattern set $P = \{p_1, p_2, ..., p_k\}$, with |P| = k where p_i denotes data structure of i^{th} line segment in P. Note that, cardinality of S and P are n and k respectively. We will need a table T with k rows (equal to $|P|$) where $T(i)$, the i^{th} row, will store some line identification numbers of line segments in S denoting the probable correspondence between the two sets.

s_factor is the scaling factor and t_vector $(\Delta x, \Delta y)$ is the translation. $s_i.slope$ gives slope of i^{th} line segment of set S and $s_i.lineno$ gives line identification number of i^{th} line segment of S.

Table T will look like following at an intermediate step,

Line segment no. of pattern set

1	i,j..
2	.
3	.
.	.
.	.
.	.
k	.

Index No. of line segments in sample set which are probable match for line segment no. 1 of pattern set on basis of slope

Finally, if match exists under similarity transformation Table T returns correspondence of unique line identification numbers from sample set S to pattern set P.

2.2. Preprocessing phase of algorithm

In this phase the sample set S is sorted on basis of slope of line segments contained in it so that now s_1 is the line segment with minimum slope and s_n is the segment with maximum slope in S. For this part the space complexity is $O(n)$ and the time complexity is $O(n\ log(n))$.

2.3 Query processing

At this stage we have the sample set ordered on basis of slope of line segments in it. The algorithm begins by calculating the possible rotations that can be given to pattern set to make its line segments parallel to line segments in the sample set. The point to note here is that if a match exists then after rotation the first line segment in the pattern set must become parallel to one of the line segment in sample set. So, a set D (|D|=w, w<=n) is formed which is a collection of distinct difference of slope of first line segment of pattern

set with the slope of line segments in sample set. Since the sample set is ordered, the distinctness of elements in D is ensured by discarding the new entry, say i^{th} entry where 1<= i <=w, if it is equal to the $(i-1)^{th}$ entry. So, set D contains all the distinct possible rotations that can give us a matching. Now, in each pass of the algorithm an element $d \in D$ is taken and each line segment in pattern set is given a rotation of d. Thus the set E (|E|=k, *where* $e_i \in E, 1 \le i \le k$ *is the slope after rotation of the* i^{th} *line segment of* P) is formed that captures the new slopes of line segments in pattern set after giving rotation d to all of them. In a particular pass each element of E is taken one by one and on basis of equality to the slope of line segments in sample set S the table T as mentioned above is formed. If any row of T is empty the particular pass is terminated and search begins again by taking a new element from D i.e. by beginning a new pass. If all the rows of T are non-empty, then entry in T(1) give the probable matching of first line segment in P to line segment in S on basis of rotation only. The scaling factor *s_factor* and translation vector *t_vector* is calculated to match exactly the first line segment of P to the line segment of S indicated by T(1). This *s_factor* and *t_vector* is applied to all the line segments of P and then it is checked whether line segments in P become exactly equal to the probable line segment in S as indicated by entry in table T. If all the line segment matches, the match is reported or else the present pass is terminated and search is carried with a new pass.

Algorithm

1 form set D = $\{d_1, d_2, ..., d_w\}$ where $d_i = s_i.slope - p_1.slope$ and each d_w is distinct ($k \le w \le n$).

2 for i ← 1 to w
do

2.1 form set E = $\{e_1, e_2, ..., e_k\}$ where $e_i = (p_1.slope + d_i) \bmod \pi$

2.2 for j ← 1 to k
do

2.2.1 if(e_j == s_m .*slope* for some $m \in \{1,2,...,n\}$)

2.2.2 then add s_m .*lineno* to $T(j)$

2.2.3 else initialize T
break inner for loop (i.e. go to 2)

2.3 while (there are entries left in $T(1)$)
do

2.3.1 find scaling factor *s_factor* to make the length of p_1 and the length of line segment in S corresponding to the current entry of $T(1)$ under consideration equal.

2.3.2 apply *s_factor* to all the line segments in P

2.3.3 find translation vector *t_vector* which translates p_1 exactly on to the line segment in S corresponding the current entry of $T(1)$ under consideration.

2.3.4 apply *t_vector* to each line segments in P

2.3.5 for r ← 2 to k
do
while(there are entries in T(r))
do
if(co-ordinates of p_r and current
entry of T(r) matches)
then go to 2.3.5
else go to 2

2.3.6 report that match is successful and print content of T, s_factor, t_vector, d_i (rotation).

2.4 initialize T and go to 2

2.5 if no match reported till now then matching does not exists.

2.3.1 Query processing complexity analysis

Step 1 of the algorithm involves forming of set D by subtracting the slope of first line segment of P from slope of each line segment of S insuring that each entry in D is distinct . Since S is sorted this will take $O(n)$ time.

The worst case of the proposed algorithm is encountered when we consider that some line segments in sample set S are parallel to each other.

So suppose S consists of l_1*line segmets of slope* m_1, l_2 *line segmets of slope* m_2, ……, l_w*line segmets of slope* m_w such that $\sum_{i=1}^{w} l_i = n$ i.e. the set S is the union of w subsets of parallel line segments and these subsets are of cardinalities $l_1, l_2, \ldots\ldots, l_w$ respectively. So that |D| = w hence the outer 'for' loop beginning at step 2 will run for 'w' times. Formation of table T in step 2.2 will take $O(k\log(n))$ time (each search for equality in 2.2.1 takes $O(\log(n))$ time as set S is ordered and this equality is checked for each of the k elements of set E).

The outer 'for' loop of step 2 will run 'w' times but 'while' loop of step 2.3 will run different number of times for each run of step 2. We can calculate complexity of step 2.3 by adding complexity of 'while' loop for each run of step 2. In the outer 'for' loop each time a $d \in D$ is chosen and each line segment in P is given a rotation of d. Therefore the first line of set P in each run of the outer 'for' loop becomes parallel to a distinct subset (as each element of D is distinct) of the set S as described above. So T(1) will have l_j entries where $l_j \in \{l_1, l_2, \ldots\ldots, l_w\}$ in each run of outer for loop with entries being distinct in any two runs. So if in a particular run of outer for loop T(1) contains l_j entries where $l_j \in \{l_1, l_2, \ldots\ldots, l_w\}$, then T(2) to T(k) will contain some number of entries from the set

$\{l_1, l_2, \ldots\ldots, l_w\}/l_j$. So for a particular run of outer for loop the time taken in the 'while' loop of 2.3 in which T(1) contained $l_j \in \{l_1, l_2, \ldots\ldots, l_w\}$, entries is given by

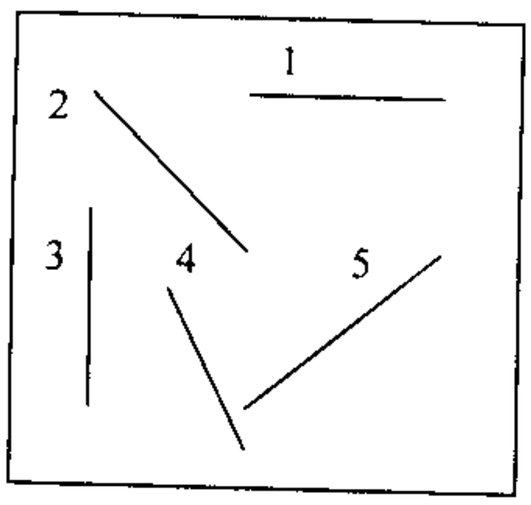
l_j * (sum of number of entries in T(2) to T(k))

which is bounded above by (l_j * n) (as $\sum_{i=1}^{w} l_i = n$). So the total time taken in step 2.3 (for all runs of outer 'for' loop) is given by $\sum_{i=1}^{w} l_i * n$ which is $O(n^2)$.

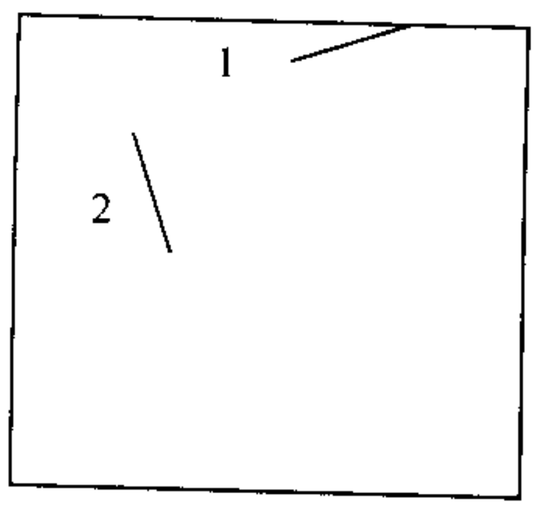
So total complexity of step 2 becomes $O(kn\log(n) + n^2)$. This also becomes the time complexity of query processing as step 1 takes O(n) time.

It can be noted that if set S consists of non parallel lines only then for any run of step 2, each row of table T will have a single entry. So the time complexity for such a case will reduce to $O(kn\log(n) + kn)$ which is $O(kn\log(n))$.

2.3.2 A Running Example



Sample set

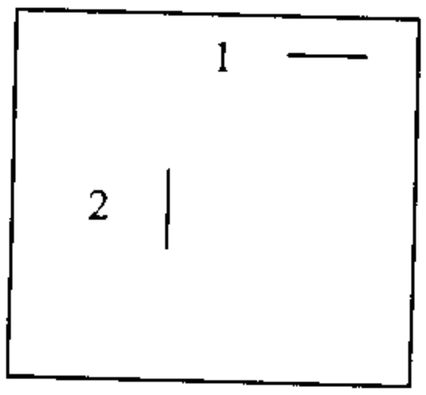


Pattern set

After calculating difference set and applying rotation on basis of that, for one of the rotation we will get table T like

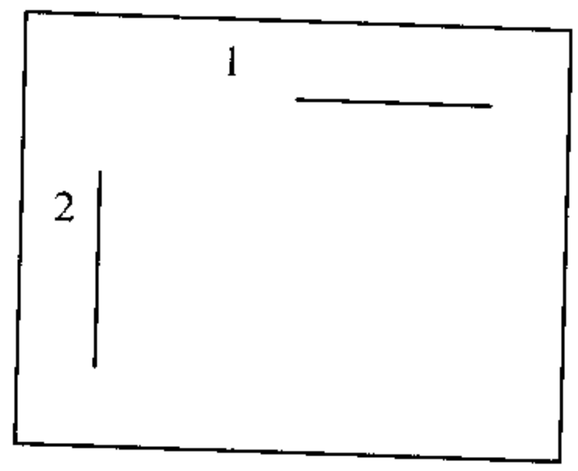
1	1
2	3

and for it pattern set will become like



then according to row 1 of table T, for matching of line segment no. 1 of pattern set to line segment no. 1 of sample set we calculate scaling factor *s_factor* and translation vector *t_vector*, and apply it to whole table.

So, now pattern set becomes like



which is clearly subset of sample set where line no. 1 of transformed pattern set matches to line no. 1 of sample set and line no. 2 matches with line no. 3.

So this way algorithm detects all possible subset matching of pattern set P in sample set S under similarity transformations.

Chapter 3

Experimental Results

In this work, we have developed an efficient algorithm for the matching of line segments in 2D under similarity transformations. Here a number of sample set S of line segments are given. For given an arbitrary pattern set P of line segments, our objective is to report one of the subset of sample set S which matches pattern set P under similarity transformations.

We have implemented our algorithm on SUN, Ultra-Enterprise, sparc; the OS version being Solaris 5.8. Table 1 shows the theoretical complexity results and Table 2 illustrates the experimental performance. We have generated random data files for sample set S. For pattern set P we pick a file randomly from sample set S and we take random subset of that file. Then we give some random transformations to it. Now we look for match for this pattern set P in sampel set S. We have generated 50 files of S for each experiment.

TABLE 1
Worst case complexity analysis

Preprocessing time	Query time	space
$O(n\ log(n))$	$O(kn\log(n)+n^2)$	$O(n)$

TABLE 2
Experimental result with respect to CPU time

# of line segments in S per file	# of line segments in P	CPU time in μ sec incuding I/O time
50	10	1023031
100	20	1086556
150	30	1130181
200	40	1203266

Bibliography

[1] A.V. Aho, J. E. Hopcroft, and J. D. Ullman [1974], *The Design and Analysis of Computer Algorithms,* Addition-Wesley,Reading, Mass.

[2] M. Overmars, M. D. Berg, M. V. Kreveld, O. Schwarzkopf [2000], *Computational Geometry: Algorithm and Applications.* Springer-Verlag, Germany.

[3] Arijit Bishnu, Sandip Das, Subhas C. Nandy and Bhargab B. Bhattacharya, *An Improved Algorithm for Point Set Pettern Matching under Rigid Motion,* ISI Kolkata.

[4] F. P. Preparata and M. I. Shamos, *Computational Geometry,* Springer-Verlag, New York, 1985.

[5] P. K. Agarwal and M. Sharir, *Efficient Algorithm for Geometric Optimization,* ACM Computing Surveys, Vol. 30, No. 4, pp. 412-458, 1998.

[6] H. Alt and L. J. Guibas, *Discrete Geometric Shapes: Matching, Interpolation, and Approximation – A Survey,* Technical Report B 96-11, Freie University Berlin, 1996.

[7] M. T. Goodrich, J. S. B. Mitchell, and M.W. Orletsky, *Appropxiate Geometric Pattern Matching under Rigid Motion,* IEEE Trans. PAMI, vol. 21, no. 4, pp. 371-379, April 1999.

[8] D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching,* A-ddision-wesley Publishing Company, Inc., 1973, Reading, Mass.

[9] P. J. Rezende and D. T. Lee, *Point Set Pattern Matching in d-dimensions,* Algorith-mica, vol. 13, pp. 387-404, 1995.

[10] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithm,* Printice Hall of India, 1998, New Delhi.

[11] Remco C. Veltcamp and Michiel Hagedoorn, *State-of-the-Art in Shape Matching,* Utrech Univesity, Netherland, 2001.