AD-A159 625

Optimal Algorithms for Finding the Symmetries of a Planar Point Set

P. T. Highnam

CMU-RI-TR-85-13

Carnegie-Mellon University

The Robotics Institute

THE ROBOTICS INSTITUTE

Technical Report

DTIC ELECTE SEP 27 1985 B

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

85 9 27 026

Optimal Algorithms for Finding the Symmetries of a Planar Point Set

P. T. Highnam

CMU-RI-TR-85-13

Department of Computer Science
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

August 1985

Copyright © 1985 Carnegie-Mellon University

This work was sponsored in part by the Defense Advanced Research Projects Agency (DoD) ARPA Order No. 3597, Amendment No. 18, monitored by Air Force Wright Aeronautical Laboratories Avionics Laboratory under Contract Number F33615-83-C-1023.

DTIC
ELECTE
SEP 27 1985
S D
B

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER CMU-RI-TR-85-13	2. GOVT ACCESSION NO. AD-A159625	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Optimal Algorithms for Finding the Symmetries of a Planar Point Set		5. TYPE OF REPORT & PERIOD COVERED Interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) P. T. Highnam		8. CONTRACT OR GRANT NUMBER(s) DoD Arpa Order No. 3597, no.18 F33615-83-C-1023
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University The Robotics Institute Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency		12. REPORT DATE August 1985
		13. NUMBER OF PAGES 6
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Air Force Wright Aeronautical Laboratories Avionics Laboratory		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release; distribution unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

This document

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We present an asymptotically optimal algorithm to locate all the axes of mirror symmetry of a planar point set. The algorithm was derived by reducing the 2-D symmetry problem to linear pattern-matching. Optimal algorithms for finding rotational symmetries and deciding whether a point symmetry exists are also presented. See p 1

Table of Contents

Accession For
NTIS GRA&I ✓
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

DTIC COPY INSPECTED 3

Abstract

We present an asymptotically optimal algorithm to locate all the axes of mirror symmetry of a planar point set. The algorithm was derived by reducing the 2-D symmetry problem to linear pattern-matching. Optimal algorithms for finding rotational symmetries and deciding whether a point symmetry exists are also presented.

1. Introduction

A set of points P in the plane has an axis of mirror symmetry A when for every point p of P not lying on A there is another point p' in P s.t. A is the perpendicular bisector of the line pp'. P has a rotational symmetry of α when rotating P about its centroid C by α is an identity operation on P. P has a point symmetry (which must be at C) precisely when P has a rotational symmetry of π.

This note presents an optimal O(n log n) algorithm for discovering all the mirror symmetries of an n point set P in detail and describes the changes needed to detect the rotational symmetries (and hence the existence of a point symmetry). Lower bounds are shown for each problem. Mirror symmetries are the objects of interest until section 5.

The 2-D mirror symmetry problem is reduced to a 1-D pattern-matching problem for which fast solutions are well-known. Any A must pass through the centroid C of P, so the points are first translated so that (0,0) corresponds to C. After expressing the points in polar coordinates, sort them increasing-distance-within-increasing angle from a reference direction. For each unique angle (at most n), replace the set of points residing at that angle (i.e., $\geq$ 1 point) by a *tuple* which is simply a list of their distance components. Let the number of unique angles be m ($\leq$ n). Consider the result as a length 2m string F, the symbols of which are alternately tuples and the angles between adjacent tuples. The mirror symmetries of P correspond exactly to the length 2m subsequences of FF which are palindromes.

The palindromes can be discovered by a fast one-dimensional string matching algorithm, looking for occurrences of the reversal of F in FF. One such algorithm is that of Knuth, Morris and Pratt [1] ("KMP") which permits the detection of all occurrences of a *pattern* within a *text* in time proportional to the sum of the lengths of the text and pattern.

Section 2 of this report contains the algorithm, and section 3 contains proof of why the algorithm works. In section 4 we demonstrate that it is impossible to improve the asymptotic time bound. In section 5 we present a variant of the basic algorithm which detects the rotational symmetries and point symmetry, and a proof that these are also optimal.

2. Algorithm

Input : n point planar point set P, (p_i for i =0(1)n-1).

Output : The centroid of P and the orientations of each of its axes of mirror symmetry.

1. Find the centroid C of P. Translate the point set so that the origin coincides with C.

2. Select an arbitrary reference direction (for convenience we choose the direction of p_0). Represent the points in polar coordinates with the angle component as measured anticlockwise from the reference direction, denote point p_i as (r_i, θ_i).

3. Sort the points by increasing-distance-within-increasing-angle. Delete any points that have zero distance ($r_i = 0$). Let the number of different angles be $m \le n$.

4. For each of the m different angles, represent its set of points in a single *tuple* , which simply holds the set of distances at that angle. Beginning at the reference direction, proceed through the list of tuples in order of increasing angle, generating the length 2m string F : at the current position append the tuple to the string; move to the next tuple, appending the angle traversed to the string. Finish when returning to the reference direction. The first element of this string is a tuple, angles and tuples alternate within it. Create the length 2m string R and the length 4m-2 string F' as shown below:

$$F = f_0 f_1 \cdots f_{2m-1}$$

$$R = f_0 f_{2m-1} f_{2m-2} \cdots f_1$$

$$F' = f_0 f_1 \cdots f_{2m-1} f_0 f_1 \cdots f_{2m-3}$$

5. Employ a string-matching algorithm such as KMP to locate all of the matches of R in F' . From the definition of F the only matches are possible beginning at even indices (since f_0 is a tuple). Denote the size t ($0 \le t \le m$) list of indices of F' at which a match can begin by I_j (j =0(1) t-1).

6. Compute for each match index I_j the orientation of an axis of mirror symmetry A_j. Let $k_j = I_j / 2$,

k_j is even => A_j passes through tuple f_{k_j}.

k_j is odd => A_j bisects angle f_{k_j}.

3. Details

In this section we demonstrate that there is a 1:1 mapping between the axes of mirror symmetry of the point set and the successful matches of R in F' . We also derive the formula for finding the orientation of the corresponding axis from a match.

A *palindrome* is a string of symbols with the property that in whichever direction it is read, it will give the same string, $s_0 s_1 \cdots s_{n-1} = s_{n-1} \cdots s_1 s_0$. An *i -palindrome* is a string with the property that if the symbol with index *i* is deleted and the left remaining substring is appended to the right of the right remaining substring, then the new string is a palindrome, $s_{i+1} \cdots s_{n-1} s_0 \cdots s_{i-1} = s_{i-1} \cdots s_0 s_{n-1} \cdots s_{i+1}$.

Because the tuples do not necessarily have a unique distance associated with them, we can consider the set of tuples $\{f_{2i}$, for $i = 0$ to m-1$\}$ as laid out on the perimeter of a circle (diameter unimportant, centre at the centroid of the point set), with tuples f_{2i} and f_{2i+2} separated by the angle f_{2i+1}. Tuples f_{2m-2} and f_0 are separated by angle f_{2m-1}. The axes of this representation are precisely the axes of the original point set.

The notation used is that of the algorithm of section 2. $F = f_0 f_1 \cdots f_{2m-1}$, where m is the number of tuples and an element f_i is either an angle (i is odd), or a tuple (i is even). $R = f_0 f_{2m-1} f_{2m-2} \cdots f_1$, and $F' = f_0 f_1 \cdots f_{2m-1} f_0 f_1 \cdots f_{2m-3}$. We say that R matches F' at index i when the length 2m substring of F' $f_i .. f_{2m-1} f_0 .. f_{i-1}$ matches R. For all such matches i must be even because f_0 is a tuple and the tuples all appear with even index in F.

LEMMA For any k $(0 \leq k < m)$: F is a k-palindrome *iff* R matches F' at index $2k$.

Define substrings,

$a_1 = f_{k+1} \cdots f_{2k-1}$ (Length k-1)

$a_2 = f_{2k+1} \cdots f_{2m-1}$ (Length 2m-2k-1)

$a_3 = f_0 \cdots f_{k-1}$ (Length k)

$b_1 = f_{k-1} \cdots f_1$ (Length k-1)

$b_2 = f_{2m-1} \cdots f_{2k+1}$ (Length 2m-2k-1)

$b_3 = f_{2k} \cdots f_{k+1}$ (Length k)

When F is a k-palindrome and $k < m$ we know that $f_{k+1} .. f_{k-1}$ is a palindrome. More specifically, $a_1 f_{2k} a_2 a_3 = b_1 f_0 b_2 b_3$. Therefore $f_{2k} = f_0$, $a_1 = b_1$, $a_2 = b_2$, and $a_3 = b_3$ using the substring lengths given above.

When R matches F' at index $2k$ we know $f_{2k} a_2 a_3 f_k a_1 = f_0 b_2 b_3 f_k b_1$. The equivalences here are again $f_{2k} = f_0$, $a_1 = b_1$, $a_2 = b_2$, and $a_3 = b_3$.

Because the equivalences enforced are the same in both cases the two situations are the same.

When an axis A passes through a tuple f_{2i}, then F is a $2i$-palindrome. Similarly, if A bisects the angle f_{2i+1} then F is a $(2i+1)$-palindrome.

LEMMA For any $k\,(0 \leq k < m)$: F is a k-palindrome *iff* k is even and there is an axis A passing through tuple f_k, or k is odd and there is an axis A bisecting the angle f_k.

There are two cases to consider,

- k is odd. Let $k = 2i+1$. That F is a $(2i+1)$-palindrome is precisely the condition brought about by A bisecting angle f_{2i+1}.
- k is even. Let $k = 2i$. That F is a $2i$-palindrome is precisely the condition brought about by A passing through tuple f_{2i}.

We have shown by the two lemmas that for any $k\,(0 \leq k < m)$, there is an axis A passing through tuple f_k (k is even), or an axis A bisecting angle f_k (k is odd) precisely when there is a match for R in F' at index $2k$. It remains to point out that this is sufficient to capture all axes because an axis A must cut the circle at two points π apart and in having k range from 0 to m-1 we have covered a complete semi-circle.

4. Complexity

In this section we show that the complexity of the algorithm presented in Section 2 matches that of the problem and so is optimal, i.e., the algorithm is O(n log n) and the problem is shown to be Ω(n log n). To show the latter we use a reduction of the set equivalence problem to that of deciding whether there is or is not an axis of mirror symmetry in a planar point set. This also shows that it is no harder to find all the axes of mirror symmetry than it is to find whether there are any at all.

From the description of the algorithm in section 2 we can see that it is O(n log n), and this derives from the sorting operation of *Step 3*. *Steps 1,2,4* and *6* are clearly O(n) while *Step 5* can be done in O(n) time using a string matching algorithm such as KMP. The worst-case complexity of the KMP algorithm is not affected by alphabet size.

Given two sets of n real numbers we wish to decide whether or not they are the same. This problem is known to be Ω(n log n) work.

Call the sets A and B and then proceed as follows:

1. Find the lowest value L_A in A and the lowest value L_B in B.
2. Create the set A' from A by adding the element $a - L_A$ to A' for each a in A. Similarly create B'

from B using L_B.

3. Manufacture the 2D point set P defined as $\{(a, a) : a \in A'\} \cup \{(b, -b) : b \in B'\}$.

4. Find the axes of mirror symmetry of the set P. If P has an axis of mirror symmetry then A and B are the same, else they are not.

The set P defined above will have at most one axis of mirror symmetry and this if present will correspond to the 'x-axis' of the 2D system created.

5. Rotational and point symmetries

In this section we present a version of the algorithm of section 2 which will find all the rotational symmetries of point set P. Hence, as noted in the introduction, we will also have discovered whether the set has a point-symmetry because that is identical to a rotational symmetry of π.

A rotational symmetry is an angle α s.t. rotation of P by α about its centroid is an identity operation on P. Furthermore, if α_0 is the smallest such angle then the set of angles of rotational symmetry is precisely the set of integer multiples of α_0. Hence, it is only necessary to find α_0.

Given the string representation F of P, proceed by matching F against F', starting with the first element of F at f_2 and moving to the right. This avoids a meaningless match of F with itself as prefix of F' and because angle α_0 corresponds to the lowest index at which F will match F'. Stop when the first match is found (or a failure occurred at the end of F', implying no rotations). Let the index of the match in F' be i, which must be even as before. The corresponding α_0 is the sum of the angles in F' lying to the left of the match, $\alpha_0 = f_1 + f_3 + ... f_{i-1}$.

The complexity of this algorithm is the same as that of the mirror symmetry algorithm, O(n log n). Checking whether α_0 is an integer divisor of π is constant time work and so O(n log n) applies to the point symmetry decision too.

We now show that deciding whether P is point-symmetric is Ω(n log n) and so the algorithm above for rotational symmetry and point symmetry is optimal. As in section 4, use the problem of deciding whether the two sets A and B are identical. We show that this is no harder than deciding if P is point-symmetric by the following algorithm:

1. Find the lowest value L_A in A and the lowest value L_B in B.

2. Create the set A' from A by adding the element $a - L_A$ to A' for each a in A. Similarly create B' from B using L_B.

3. Manufacture the 2-D point set P defined as $\{(a, a) : a \in A'\} \cup \{(-b, -b) : b \in B'\}$.

4. Decide whether P is point-symmetric, if it is then A and B are the same, else they are not.

6. Acknowledgments

Dan Hoey provided helpful discussion at the right time. It's been a very long time since this was written and several people have made comments which have improved the presentation particularly G.J. Agin.

References

[1] D.E.Knuth,James H.Morris Jr. and V.R.Pratt, Fast Pattern Matching in Strings, *SIAM J COMPUT 6*, 2 (June 1977), 323-350.