THE ASTRONOMICAL JOURNAL VOLUME 91, NUMBER 5 MAY 1986

A PATTERN-MATCHING ALGORITHM FOR TWO-DIMENSIONAL COORDINATE LISTS

EDWARD J. GROTH
Physics Department, Jadwin Hall, Princeton University, Princeton, New Jersey 08544

Received 31 December 1985; revised 28 January 1986

ABSTRACT

A pattern-matching algorithm for two-dimensional coordinate lists is described. The algorithm matches pairs of coordinates in two lists based on the triangles that can be formed from triplets of points in each list. The algorithm is insensitive to coordinate translation, rotation, magnification, or inversion and can tolerate random errors or distortions.

I. INTRODUCTION

A problem that arises frequently involves matching points found in two lists of two-dimensional coordinates where the coordinate systems are not the same and the matching must be based on the identification of similar geometrical configurations of points in both lists. An algorithm is described that will accomplish such matching automatically, provided the two lists have a sufficient number of points in common, the distortion between the coordinate systems is not too severe, and the random coordinate errors are not too large. The algorithm is completely insensitive to any translation, rotation, magnification, or inversion between the two coordinate systems. The algorithm was developed to solve a particular problem in astronomical data analysis but should have application to a variety of problems in image processing.

The astronomical problem involves the determination of stellar positions. From a photograph of a region of the sky, the positions of a number of stars are measured. Of course, these coordinates are in the system of the measuring engine. The celestial coordinates (equivalent to latitude and longitude) of the brighter stars are listed in a catalog. By identifying the measured stars with the catalog stars, the transformation from the measuring-engine system to the celestial system can be determined and used to calculate celestial coordinates for all measured stars. The algorithm described below performs the identification step and is similar to the actions performed by an astronomer when attempting to visually match two photographs of the same region of the sky. The astronomer looks for similar triangles among the bright stars.

The objective of the algorithm is *not* to match all points in two lists of arbitrary size. If this is the application, as it is in the astronomical example mentioned above, the algorithm is applied to a subset ($\sim$20 points) of each list. From the matches found, a coordinate transformation is derived and used to place both lists in the same coordinate system. Other points in the lists can then be matched by conventional techniques, e.g., by matching points that are sufficiently close.

In Sec. II below, the algorithm is described, while in Sec. III some examples of its application are presented. These examples serve to illustrate and quantify some of the statements made in Sec. II.

II. THE ALGORITHM

The algorithm contains several major steps: selecting the points to be matched, generating lists of triangles, matching the triangles, reducing the number of false matches, assigning matched points, and protecting against spurious assignments. Each of these is described in turn.

a) Selecting The Points To Be Matched

The first steps in the algorithm involve constructing the two lists that are to be matched. The algorithm provides for a tolerance ϵ in the precision to which points are matched. If two points in a list are closer than ϵ, then either could potentially be matched with the same point in the other list. This is clearly a confusing situation and all pairs closer than an elimination threshold ξ are eliminated from each list. Both ϵ and ξ are user-specified parameters, but typical defaults are $\epsilon = 0.001$ and $\xi = 3\epsilon$, where ϵ has been expressed in units such that the coordinates occupy the interval $[0,1]$ in each dimension. This step also eliminates duplicates which would lead to divide-by-zero errors if allowed to remain.

As will be seen below, some steps in the algorithm require computation time proportional to a high power of n, the number of elements in either list, so it is important to reduce the size of each list. For a typical minicomputer, n in the range 20–30 is a manageable number. In the case of the astronomical problem, the n brightest stars in each list can be selected. For other applications, other selection criteria may be appropriate. With some applications, it is possible that lists of unequal length may provide a higher probability of obtaining matches. The algorithm does not require that the lists be of equal length, but the following description and example are based on lists of the same length.

b) Generating Triangle Lists

Once the lists have been constructed, a list of triangles is generated for each list of points. Initially, the triangle lists include all possible triangles formed from any three points as vertices and contain the following information for each triangle: the three points forming the vertices arranged so that the side between vertices 1 and 2 is the shortest side, the side between vertices 2 and 3 is the intermediate side, and the side between vertices 3 and 1 is the longest side; the logarithm of the perimeter of the triangle; the orientation—whether vertices 1, 2, and 3 are traversed in a clockwise or counterclockwise sense; the ratio of the longest side to the shortest side; the tolerance in the ratio; the cosine of the angle at vertex 1; and the tolerance in the cosine.

The angle at vertex 1 could be used in place of the cosine at vertex 1. If the angle were measured in a definite sense such as counterclockwise from the shortest to the longest side, it would carry the information about the sense of vertex traversal. In order that the algorithm be independent of coordinate inversion, the angle comparisons would have to be made after folding the angles into the interval 0°–180°. Determining the sense of a triangle match (Sec. *c* below) would

0004-6256/86/051244-05$00.90

require numeric comparison of angles rather than logical comparison of the directions of vertex traversal. The evaluation of the cosine and its tolerance [Equations (5) and (6) below] requires nothing more complicated than a square root, while trigonometric functions would be required to deal with the angle. For these reasons, the algorithm design is based on the sense of vertex traversal and the cosine of the angle at vertex 1, rather than the angle itself.

The tolerances in the ratios and cosines are computed by treating the matching tolerance ϵ as an independent error in each coordinate and propagating these errors through the expressions for the ratio and cosine. In particular, if (x_1,y_1), (x_2,y_2), and (x_3,y_3) are the coordinates of vertices 1, 2, and 3, then the ratio is given by

$$R = r_3/r_2, \tag{1}$$

where

$$r_3 = \sqrt{\Delta x_3^2 + \Delta y_3^2},\ \Delta x_3 = x_3 - x_1,\ \Delta y_3 = y_3 - y_1, \tag{2}$$

$$r_2 = \sqrt{\Delta x_2^2 + \Delta y_2^2},\ \Delta x_2 = x_2 - x_1,\ \Delta y_2 = y_2 - y_1, \tag{3}$$

and the tolerance in R is

$$t_R^2 = 2R^2\epsilon^2\left(\frac{1}{r_3^2} - \frac{C}{r_3 r_2} + \frac{1}{r_2^2}\right), \tag{4}$$

where C is the cosine of the angle at vertex 1. C and its tolerance are given by

$$C = (\Delta x_3 \Delta x_2 + \Delta y_3 \Delta y_2)/r_3 r_2, \tag{5}$$

$$t_C^2 = 2S^2\epsilon^2\left(\frac{1}{r_3^2} - \frac{C}{r_3 r_2} + \frac{1}{r_2^2}\right) + 3C^2\epsilon^4\left(\frac{1}{r_3^2} - \frac{C}{r_3 r_2} + \frac{1}{r_2^2}\right)^2, \tag{6}$$

where S is the sine of the angle at vertex 1.

Triangles whose ratio is too large are discarded from the lists. The ratio limit is a user-specified parameter; a typical limit is 10. The reason for this step is that a triangle with a large ratio often has a large ratio tolerance produced by a very close pair of vertices. As a consequence, many triangles in the other list must be examined in order to find a match, and even when a match is found, it is not likely to be a reliable match. With a ratio limit of 10, the number of triangles is reduced by a few percent, while the decrease in computation time due to the reduction in the number of triangles that must be searched is about a factor of 2.

c) Matching Triangles

The two lists of triangles are compared and matching triangles are identified whenever

$$(R_A - R_B)^2 < t_{RA}^2 + t_{RB}^2, \tag{7}$$

and

$$(C_A - C_B)^2 < t_{CA}^2 + t_{CB}^2, \tag{8}$$

where the subscripts A and B refer to the two lists. In the event that more than one triangle in list B matches a triangle in list A according to criteria (7) and (8), the closest is chosen. By construction, this procedure generates the same list of matched triangles independent of any translation, rotation, magnification, or inversion between the A and B coordinate systems.

Matching is accomplished with a procedure similar to those used in sort-merge algorithms. Both lists are sorted in order of increasing ratio and the maximum ratio tolerance is determined for each list. Successive triangles are selected from list A and compared with a range of triangles from list B. Since the lists are sorted, the end of the range (for a given triangle from list A) occurs when an R_B is encountered that is too large to satisfy criterion (7) even when the maximum tolerances are used. Similarly, the start of the range is updated (for the next triangle from list A) to the first R_B that is just large enough to satisfy criterion (7) with the maximum tolerances. With this procedure, it is not necessary to compare every triangle in list A with every triangle in list B but only those within a range of R_B proportional to ϵ.

Triangles may be matched in the same or opposite senses; that is, if both triangles of a matched pair have clockwise or both have counterclockwise orientations, the pair have the same sense, otherwise they have the opposite sense.

The procedure generates both true and false matches. If the three pairs of matching vertices in a matched pair of triangles contain points that actually correspond, then the match is a true match, otherwise a false match. Assuming the points are randomly distributed, an estimate of the number of false matches can be obtained by the following argument. If there are n points in each list, there are $n_t = n(n-1)(n-2)/6$ triangles and $12n_t^2$ potential matches (there are six ways to match the vertices and two ways to match the orientations of a pair of triangles). Consider a triangle from list A with vertices 1, 2, and 3 and a triangle from list B with vertices 1′, 2′, and 3′. Conceptually perform a coordinate transformation on the A triangle so vertices 1 and 2 line up exactly with vertices 1′ and 2′. The algorithm finds these triangles to match (for this particular arrangement of vertices and orientation) if vertex 3 is close enough to vertex 3′. The ratio and cosine tolerances are constructed by assuming a tolerance of $\pm\epsilon$ in each coordinate, so it is to be expected that the triangles match if vertex 3 falls within a square of side 2ϵ centered on vertex 3′. The probability of this event is $4\epsilon^2$, so the average total number of false matches is $48n_t^2\epsilon^2$. This is an approximate calculation and two effects have been ignored: edge effects and the fact that vertices 1 and 1′ and vertices 2 and 2′ need match only to within the tolerance. Edge effects reduce the number of false matches slightly, while tolerances at the other vertices increase the number of matches. Both effects are difficult to calculate analytically, so the adopted expression for the number of false matches is

$$n_f = 48 f n_t^2 \epsilon^2, \tag{9}$$

where f is a factor to be determined but may be expected to be in the range 2–3. Since n_t is the upper limit to the number of matches, Equation (9) can only be applied when it predicts $n_f \ll n_t$.

Since the number of triangles is roughly proportional to n^3, it might be thought that the computation required for the triangle-matching procedure is proportional to n^6. Since not all triangles in list B are compared with all triangles in list A, the matching procedure described above requires computation proportional to $n^6\epsilon$. For fixed ϵ, the computation is still proportional to n^6. Equation (9) shows that as n increases, ϵ must decrease in proportion to $n^{-3/2}$ in order that the true matches not be swamped by the false matches, so the required computation is proportional to $n^{4.5}$.

d) Reducing The Number of False Matches

The next step is the elimination of as many false matches as possible. For this purpose, the previously computed logarithms of the perimeters are used. For a matched pair of triangles,

$$\log p_{\mathrm{A}} - \log p_{\mathrm{B}} = \log M, \tag{10}$$

where p_{A} is the perimeter of the triangle in list A, p_{B} is the perimeter in list B, and M represents the magnification of the coordinates between list B and list A. The magnifications for all the true matches are essentially the same, while those for the false matches have a broad distribution. The average $\log M$ and the standard deviation of the $\log M$ distribution are computed. Any match whose $\log M$ differs from the average by more than a factor times the standard deviation is discarded and the procedure is iterated until no more matches are discarded (the usual case), a preset iteration limit is reached, or all matches are discarded (in this case it is declared that the lists cannot be matched). The factor that multiplies the standard deviation is determined as follows. Let the number of same-sense matches (i.e., both triangles have the same orientation) be n_+ and the number of opposite-sense matches be n_-. Since all true matches must be entirely the same sense (when there is no coordinate inversion between lists A and B) or entirely the opposite sense, estimates of the number of true and false matches can be obtained from

$$m_t = |n_+ - n_-|, \tag{11}$$

$$m_f = n_+ + n_- - m_t. \tag{12}$$

Then if $m_f > m_t$, the factor is 1; if $0.1 m_t > m_f$, the factor is 3; and in between the factor is 2. This procedure begins by discarding many matches but discards very few once m_t becomes comparable with m_f. Finally, the remaining opposite-sense matches are discarded if $n_+ > n_-$, or the same-sense matches are discarded if $n_- > n_+$.

e) Assigning Matched Points

At the end of the discard step there are still false matches, so it is not possible to take all pairs of points contained in the remaining matched triangles as matched points. Instead, the matched triangles are used to "vote" for matched points. Each matched triangle casts three votes—one for each pair of vertices. After all the votes are cast, the vote array is sorted from maximum vote to minimum vote. If no pair of points received more than one vote, it is declared that the list cannot be matched. Otherwise, successive pairs of points in the sorted array are assigned as matched pairs until one of three events occurs: the vote drops by a factor of 2, an attempt is made to assign a point that has already been assigned, or the vote drops to zero. The first event is the usual way assignments terminate when true matches have been found. This arises because points that are true matches are involved in many matched triangles and receive a large number of votes. On the other hand, a pair of points that do not match are contained in only a few of the (incorrectly) matched triangles and receive a small number of votes.

f) Protecting Against Spurious Assignments

The algorithm just described works quite well when the two lists contain a reasonable number of points in common. On the other hand, the algorithm almost always finds a few matches, even when the lists have no points in common. To guard against this possibility, the entire procedure is repeated, beginning with the triangle generation step, but using only those points that were found to match in the first attempt. If fewer matches are found, then it is assumed that the original matches were false, and it is declared that the two lists cannot be matched. (This step is omitted if all points in the original lists were matched.)

III. EXAMPLES AND APPLICATIONS

The examples in this section are taken from the astronomical application mentioned in the introduction. A photograph of the sky was digitized on a 3150×3150 square grid. Positions and intensities of several thousand stars were computed from the digitized data. However, the positions are in units of digitizer pixels, and it is required to transform them to celestial coordinates. The positional accuracy for the brighter stars is about ± 0.06 pixels.

With knowledge of the approximate size and celestial coordinates of the center of the photograph, stars were selected from a catalog of bright stars with known coordinates. This yielded 165 stars. The celestial coordinates were projected onto a rectangular grid with origin at the approximate field center. The units are arcseconds and the range of "catalog" coordinates is ± 12 000 arcsec with typical accuracies of ± 0.3 arcsec.

To determine the transformation from pixel coordinates to celestial coordinates, the catalog stars are identified with data stars so a polynomial fit of data coordinates versus catalog coordinates can be performed. The pattern-matching algorithm is used to make the first identifications, after which a preliminary transformation is calculated and additional identifications are made.

Accordingly, the 25 brightest catalog and data stars were selected for matching. Since the catalog intensities and data intensities are determined in different colors it is not expected that all stars will match.

Figure 1 shows the two sets of 25 stars after the coordinate transformation has been determined. The 18 stars shown as

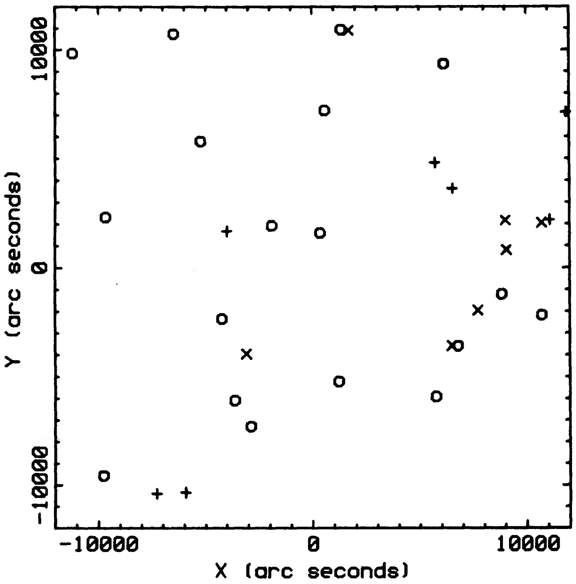

FIG. 1. Catalog and data stars used in the example. The data stars have been transformed to the catalog system. Circles indicate both a catalog and data star which the algorithm matched. Plusses (+) indicate unmatched catalog stars and crosses (×) indicate unmatched data stars.

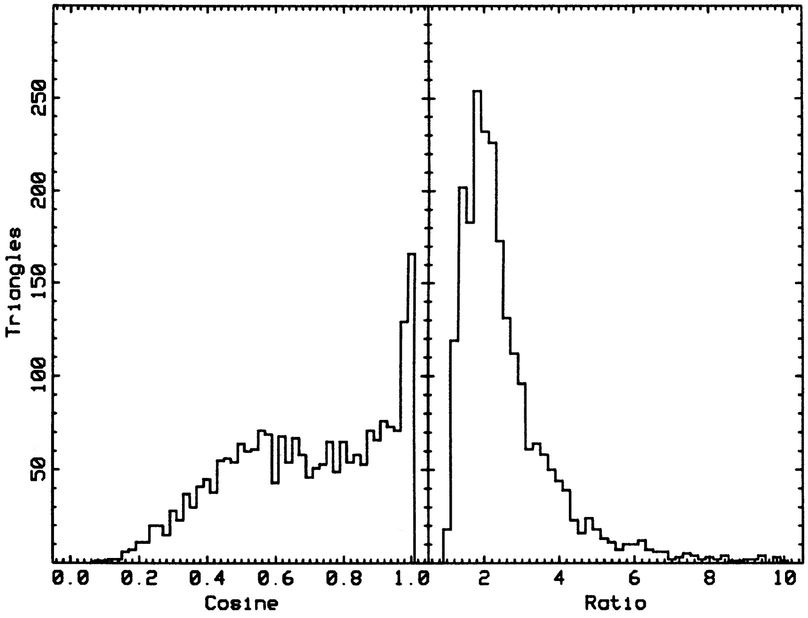

FIG. 2. Distribution of cosines and ratios of the catalog triangles for the example discussed in the text.

circles were identified by the pattern-matching algorithm. These are the only identifications possible, as all other data or catalog stars are well separated from each other.

Figure 2 shows histograms of the ratios and cosines computed for the catalog list. With 25 stars in each list there are 2300 potential triangles in each list. Since triangles with ratios greater than 10 were discarded, the algorithm actually used 2264 from the catalog list and 2216 from the data list. From Fig. 2 it appears that a smaller ratio limit would be acceptable, and indeed the algorithm finds the same matches even with a limit as low as 2. However, its performance is marginal, as some matches received barely enough votes (4) to be identified. The advantage of a lower ratio limit is a reduction of computation time in the triangle-matching step, in this case by almost a factor of 30. The setting of the ratio limit is a tradeoff between reliable performance and reduced computation time.

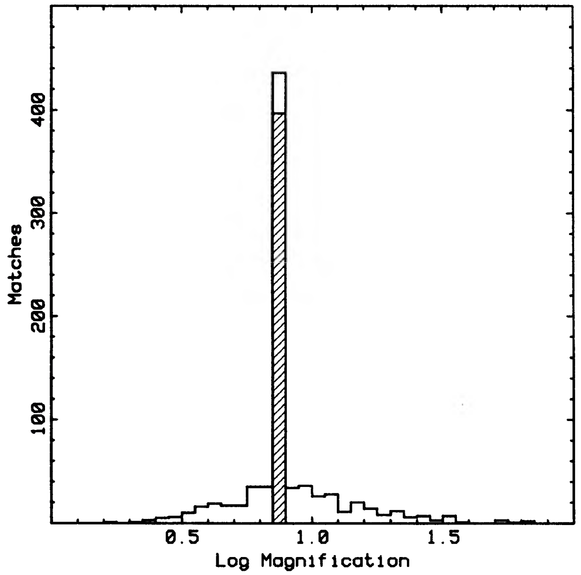

FIG. 3. Distribution of the logarithms of the magnifications of the matched triangles for the example discussed in the text. The cross-hatched histogram is the distribution remaining after false matches were eliminated.

Figure 3 shows the histogram of log M [Eq. (10)] before and after false matches were discarded. Matching found 602 same-sense and 218 opposite-sense matches. After discarding, there remained 397 same-sense matches of which three were false: discarding matches based on their magnifications is a powerful means of eliminating false matches.

Since 820 matches is not negligible compared to 2216 possible matches, the applicability of equation (9) is questionable. Nevertheless, if these data are used to estimate f, the result is 1.8, as shown in the first line of Table I. The other lines of Table I show the test number ϵ, the number of false matches, the number of true matches, and the resulting estimate for f as ϵ is varied over two orders of magnitude. At large ϵ, the number of false matches becomes comparable with the number of possible matches, resulting in small estimates of f. At small ϵ, the statistics are poor. Nevertheless, the trend in Table I allows the conclusion that $f \sim 3$ is the appropriate value for equation (9) when small numbers of false matches are predicted.

Table I also provides insight into the effects of coordinate distortions on the performance of the algorithm. (For these data, random coordinate errors are not a problem as they are always smaller than the smallest ϵ in Table I.) The photographic data contain distortions which reach 0.0017 (ex-

TABLE I. Matching as a function of ϵ.

Test	ϵ	False	True	f
1	0.001	426	394	1.8
2	0.00005	2	11	3.3
3	0.0001	7	31	2.9
4	0.0005	150	217	2.5
5	0.002	881	479	0.9
6	0.005	1774	120	0.3

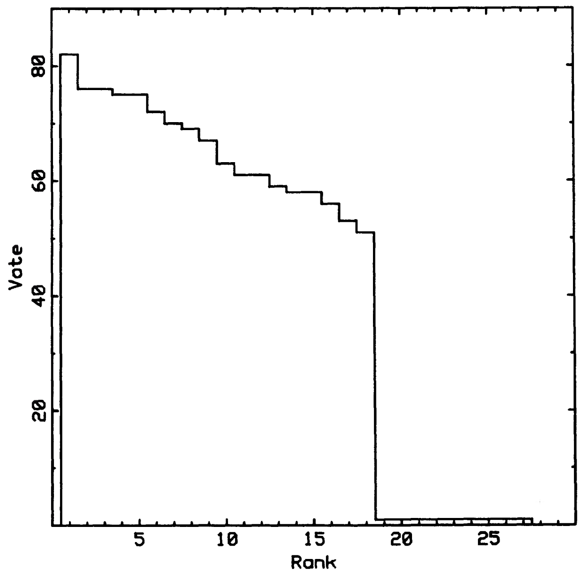

FIG. 4. Vote distribution for the example discussed in the text.

pressed in the same units as ϵ) at the edge of the field. For small ϵ, the algorithm can only match small triangles unaffected by distortion. As ϵ increases, the algorithm is less sensitive to distortion and finds more matches. However, at the largest ϵ in the table, there are so many false matches that they begin to overwhelm and hide the true matches. The choice of ϵ represents a tradeoff between reliable performance and the ability to deal with distortions.

Figure 4 shows the vote array resulting from test 1 in Table I. There were a total of 1141 votes cast, of which only nine were for incorrect pairs of points. The incorrect votes came from three incorrectly matched triangles. Figure 4 illustrates the point made in Sec. II*e* that when correct matches are found there is a very sharp transition in the vote between correct and incorrect matches.

To test the sensitivity of the algorithm to the fraction of stars that actually match, test 1 was repeated with the same conditions except that a subset of the catalog stars were redistributed at random over the area of the photograph. The algorithm found all correct matches and no incorrect matches with as few as 6 of 25 stars in common. The algorithm failed when there were only 5 of 25 stars in common. The exact point at which the algorithm fails must depend to some extent on the data and on the choice of parameters. These tests indicate that the algorithm can be expected to work reliably for lists with only 50% of the points in common and it may work with as few as 25% of the points in common.

The algorithm was tested with the same measured stars but with catalog stars selected from a different area of the sky. In this case no matches are expected and none were found. No pair of stars received more than one vote.

For the tests described in this section, the algorithm was implemented in FORTRAN on a VAX 11/750 with the VMS 3.4 operating system. Representative computation times are taken from test 1 in Table I. The times (in seconds) required for the several steps of the algorithm were: generating and sorting the triangle lists, 21.4; matching triangles, 131.7; discarding false matches, 2.6; voting 1.2; and repeating the algorithm on the matches found, 12.2.

IV. CONCLUSION

An algorithm has been presented for matching two-dimensional lists of coordinates. The algorithm has been shown to work well in its intended application and may be of use in other two-dimensional pattern-matching applications.

I thank J. R. Kuhn and the referee for critical readings of this manuscript. This research was supported in part by NASA through the Hubble Space Telescope Program contract NAS5-25084.