A Feature-Based $O(N^2)$ Approach to Point Pattern Matching

F. Murtagh*
Space Telescope – European Coordinating Facility
European Southern Observatory
Karl-Schwarzschild-Str. 2
D-8046 Garching bei München, Germany
(* Affiliated to Astrophys. Div., Space Sci. Dept., European Space Agency.)

Abstract

We describe a new algorithm for the astronomical problem of matching star lists, given by their 2-dimensional coordinates. It is successful at determining mappings between two point sets which are invariant to: translation, scaling, perturbation, random insertions and deletions, and rotation. This algorithm is feature-based, and is of $O(n^2)$ computational complexity.

1 Introduction

The problem of point pattern matching arises in 2-dimensional photometry, and in matching star lists against catalog information. The former area will be most at issue in this paper. Star lists (i.e. centroids of star positions) can arise through different point spread function (PSF) fitting procedures, or different 2-d photometry packages used on the same field; or through the reduction of images based on different color filters; or through reduction of partially-overlapping images taken at different times or with different detectors.

The term "point" (coordinates, ordinarily in 2-dimensional space) will be used interchangeably with star centroid or central location in this article. A matching of some points from one list with some points in the other list(s) is sought. Equivalently the transformation which optimally maps one list into the other is sought. This allows subsequent judgement of the methods which produced the point lists, or the data on which they were based, etc.

The next section reviews some approaches adopted for this problem, in astronomy and in computer vision generally.

1.1 A Short Review of Approaches

The matching of 2- or 3-dimensional points from two lists is a very common problem. Approaches which have been availed of include: least squares [8]; a geometric approach [1]; matching of Delaunay triangulations [5]; branch-and-bound to expedite exhaustive search [**?**]; and constraint satisfaction solved using a Hopfield-Tank neural network [6]. Many further references can be found in the works cited.

We now briefly review three approaches used in the astronomical context.

Routine `PAIR`, authored by A. Lauberts, assumes a translation between A and B. The Euclidean distance between each point i in A and each point i' in B is determined. If $i' = i + c$, then $d^2(i, i') = c^2$ for matched points, and one would expect a spread distribution of values for distances with points i'' which ought not to be matched. Thus the mode of all pairwise distances (between all i in A and all i' in B) allows the value c to be determined. This approach is invariant to translation and random perturbation.

The use of a Delaunay triangulation to capture affine-invariant information on point pattern interrelationships has been mentioned. Groth [4] implements an all-triangles matching approach. A range of speed-ups are applied to cut down on the matching of all triangles from the first list, i.e. $O(n^3)$, with a similar list from the second list. The principle efficiency tactic is to only match triangles with a ratio of longest side to shortest side which is within some tolerance. Groth finds the order of magnitude increase in computation to be improved from $O(n^6)$ to $O(n^{4.5})$. It is recommended that the number of points in both lists be limited to between 20 and 30 points for computational reasons, and such a selection may be carried out on the basis of the magnitudes associated with points.

Stetson [7], discussing an algorithm he developed many years earlier, also implemented a triangle-based matching algorithm. Points are considered in increasing order of magnitude. Following an initial matching of a small number (three) of highly-weighted points, further points are added one at a time. Hence the procedure, reasonably, is biased towards points of large weight. Imprecision in measurement of magnitudes is taken into account insofar as rank orders of magnitudes are used.

2 The Proposed Method

For each point, i, in either list, a "world view" vector is determined. This vector represents the $n-1$ other points in the same list, as "seen" from the point i. Relative to an initially arbitrary horizontal axis, the angles subtended by the $n-1$ other points to the given point, i, are determined, and sorted. We consider here, and in our experimentation, only 2-dimensional point-sets: angles could not be trivially sorted in dimensions higher than 3. At the angle subtended by j relative to i, we consider the effect of j as being related to $d(j,i)$, the usual Euclidean distance. We define the effect of j on i as $K-d(i,j)$ where K is a constant which is somewhat greater than the maximum $d(i,j)$ for all i,j. For scale independence, the value of $K-d(i,j)$ is mapped onto $[0,1]$ (by subtracting the minimum such value, and dividing by the maximum minus the minimum).

We experimented with the incorporation of magnitudes into this "effect-of-j-on-i" term, leading for example to $w_j/d^2(i,j)$ where w_j was the magnitude of star j. An alternative scheme is to use $\mid w_i - w_j \mid .(K-d(i,j))$. We currently recommend against doing this, since there can be appreciable differences in the distributions of the values of the two terms (i.e. $\mid w_i - w_j \mid$ and $K-d(i,j)$, for all pairs i,j). Consequently one or other of these terms can predominate. Standardization or normalization, per se, does not allow us to bypass this difficulty.

To summarize: with each point i in a given point-list containing in total n points, we now have have a set of "effect" terms induced by the remaining $n-1$ points. This set of "effect" terms is ordered by the angle between j and an arbitrary axis through i. Without loss of generality, this arbitrary axis may be taken as parallel to the x-axis of the given coordinate values, and the ordering may be determined in a counter-clockwise fashion. The "world view" of point i is thus expressed by this ordered set of $n-1$ values. Matching will later be carried out by seeking a point, or points, in the second list with a similar "world view".

The "world view" list of i may be interpreted as the set of projections (defined in a particular way) of $n-1$ points onto the unit circle of center i.

Any "world view" in point set A is an ordered $(n-1)$-list. Any "world view" in point set B is an ordered $(m-1)$-list. Optimally matching vectors of differing lengths can be carried out using dynamic programming. We adopted a different approach. Given that 2-dimensional data is under consideration, the angles necessarily lie between 0^o and 360^o. Hence, instead of using the given ordered list, we map this into a list of length 360 corresponding to the "world view" of a point sampled at 1^o intervals. The choice of 1^o intervals is quite arbitrary. It was found to offer a good compromise between sensitivity and computational cost. Experiments with of the order of 20 to 70 points in two lists, using 10^o intervals, were also successful.

To determine the "world view" at a given angle (at an interval of 1^o from the previous and subsequent angles), we interpolated from values of p at the angles which were larger and smaller. Note that we must allow for the fact that angles mod 360 are used. Linear interpolation was used: it was simple to implement, and gave satisfactory results.

Rebinning the "world view" vectors in this manner was carried out to allow the use of the usual Euclidean distance between the new (360-valued) vectors. There are pitfalls here: *linear* rebinning does not necessarily take a continuous "world view" into account; rebinning to 1^o intervals may not be appropriate for dense point-sets; more awkwardly such bin sizes may well be problematic for closely-packed points in a given point-set. Although the simple approach adopted worked well, it is clear that further study of these issues could be profitable.

As is clear from the foregoing, particular implementation choices were made in a number of instances.

The "world view" of any point, expressed as an ordered 360-list, is now directly comparable irrespective of what point-set the point came from. The (360-valued) "world view" vector of points is now compared to the "world view" vectors of all points in the second point-set. The usual Euclidean distance is used. Point i is *matched* with a point from the second point-set when the corresponding "world view" vectors have minimum Euclidean distance.

This minimum Euclidean distance can be used as a measure of how good the match is, since it represents how similar the "world views" are. To facilitate interpretation, such match values are discretized to a [1, 10] confidence scale. This allows the results of the matching to be expressed as: "Point i from the first list is mapped onto point i' from the second list, with confidence 4", for example. Only matches above some user-specified confidence threshold, which correspond to small distances between "world view" profiles, are used to determine an expression for the overall transformation between A and B. A threshold confidence of 3 (i.e. confidences $= 1, 2$, or 3) was found to perform well.

Even within these high-confidence matches, there can be discrepancies. An average high-confidence transformation could be determined, which takes A into B. We instead favored a robust estimate, and found the median of these high-confidence values to provide satisfactory results.

Rotation is incorporated into this algorithm as follows. We consider all possible matchings between A, and 360 versions of B: i.e. the "world view" vectors of B would be all together rotated by 1^o in successive versions. We would seek a best matching from the 360 results. Computationally, this implies 360 runs of the above algorithm. If the user knows the approximate angle of rotation, then some restricted angle-interval, alone, can be searched. We have experimented with all 360 rotations of point-set B vis à vis point-set A, and also restricted (e.g. 10 degree) intervals, and results were quite conclusive in all cases. In our experimentation, we have generally found 80–90% of matches to indicate a unique rotation-angle. A lower threshold of, e.g., 30% is currently used to signal a lack-of-consensus situation, and hence unmatchable point-sets.

As currently implemented, we have not catered for "flipping" of points, i.e. reflection in an arbitrary axis. A solution to this could be based on reversing the order of the "world view" vector values of one of the point-sets. If $O(n)$ points are provided in either point-set, our approach requires $O(n^2)$ time to determine the "world views" of all points; and subsequently $O(n^2)$ time to carry out the matching. Storage is seen to be $O(n^2)$. Some further details of this algorithm can be found in [3].

3 Example

Figures 1 and 2 show two point-sets to be matched (data courtesy of S. Ortolani). A magnitude-limit of 14.0 yielded the point-sets shown in Figures 3 and 4. The magnitude of 14.0 is arbitrary, with the sole requirement that around 100 points (a compromise between many points, leading to a robust solution, versus computational expense) should result in either list. The first few results of the matching are as follows. They are given as a triple: point sequence number in first figure, point sequence number in second, and confidence factor (1 best, 10 worst):

(1, 80, 4), (2, 54, 2), (3, 16, 2), (4, 57, 1), (5, 59, 1), (6, 111, 2), (7, 4, 1), (8, 61, 1), (9, 22, 1), (10, 65, 1).

Figure 4 was rotated by 25^o clockwise: see Figure 5. A matching between Figures 3 and 5 therefore used the potential of our algorithm for handling rotation.

Using the transformation yielded by the feature-based algorithm on magnitude-limited point-sets, the full matching of all points shown in Figures 1 and 2 was carried out. We use a rough measure of acceptable correspondences as a matched distance of less than 1.0. Using this measure, we find that 1685 points are matched from 1883 points in Figure 1 and 2552 points in Figure 2.

A range of cleverer approaches for best match searching in two dimensions (some of which are reviewed in chapter 2 of [3]) would considerably speed up this phase of the processing. It is probably the case that the feature-based phase of the processing could be made more efficient. Further enhancement of the algorithm could also handle reflection in an axis.

Sample timings of the method implemented are as follows. Feature-based matching for about 100 points in both point-sets, without rotation, requires about 25 seconds CPU time on a SPARCstation 2. For rotation, about 14 seconds per degree checked out is required. Given the rotation angle and the translation, a full matching of the data shown in Figures 1 and 2 (comprising about 2000 points in the two point sets) takes about 18 seconds CPU time on a SPARCstation 2. In this latter case, a brute-force, unintelligent best match algorithm (i.e. $O(n^2)$) was implemented.

Acknowledgements

Motivation for the approach described here arose from discussions with S. Ortolani, P.B. Stetson, and E.J. Groth, to whom I am grateful.

References

[1] P.M. Griffin and C. Alexopoulos, "Point Pattern Matching using Centroid Bounding", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, pp. 1274–1276, 1989.

[2] F. Murtagh, "A New Approach to Point Pattern Matching", *Publications of the Astronomical Society of the Pacific*, 1992, in press.

[3] F. Murtagh, *Multidimensional Clustering Algorithms*, Physica-Verlag, Wuerzburg, 1985.

[4] E.J. Groth, "A Pattern-Matching Algorithm for Two-Dimensional Coordinate Lists", *The Astronomical Journal*, Vol. 91, pp. 1244–1248, 1986.

[5] H. Ogawa, "Labeled Point Pattern Matching by Delaunay Triangulation and Maximal Cliques", *Pattern Recognition*, Vol. 19, pp. 35–40, 1986.

[6] B. Parvin and G. Medioni, "A Constraint Satisfaction Network for Matching 3D Objects", Proc. IJCNN, Vol. II, pp. 281–286, 1989.

[7] P.B. Stetson, "The Techniques of Least Squares and Stellar Photometry with CCDs", preprint, Dominion Astrophysical Observatory, 1990, 83 pp.

[8] S. Umeyama, "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, pp. 376–380, 1991.