Short Papers

Approximate Geometric Pattern Matching Under Rigid Motions

Michael T. Goodrich, *Senior Member*, *IEEE*,

Joseph S.B. Mitchell, and Mark W. Orletsky

Abstract—We present techniques for matching point-sets in two and three dimensions under rigid-body transformations. We prove bounds on the worst-case performance of these algorithms to be within a small constant factor of optimal and conduct experiments to show that the average performance of these matching algorithms is often better than that predicted by the worst-case bounds.

Index Terms—Hausdorff distance, pattern matching, registration.

———————————— ✦ ————————————

1 INTRODUCTION

SUPPOSE we are given a set B of n points in $\mathbb{R}^d$, which we shall call the *background*, and a set P of m points in $\mathbb{R}^d$, which we shall call the *pattern*. The *geometric pattern matching* problem is to determine a rigid motion, taken from some class of motions, such that each point in P is moved to a point in B.

It is easy to solve this problem if we insist on *exactly* matching points in P to points in B: Store B in a dictionary, designate one point of P as a "reference point," and consider the n placements of P corresponding to the reference point coinciding with each point of B; for each placement, do $m - 1$ queries into the dictionary to determine if all m points of P are matched. Unfortunately, this approach is very sensitive to noise. Thus, it is more natural to pose the *approximate geometric pattern matching* problem: Find a rigid motion of P such that each point of P is moved *near* to a point in B. Formally, we desire a rigid motion T, taken from some class of motions C, such that the directed Hausdorff distance from $T(P)$ to B is minimized. Recall that the directed Hausdorff distance,[1] $h(C, D)$, from a point set C to another point set D is defined as $h(C, D) = \max_{c \in C} \min_{d \in D} \rho(c, d)$, where ρ is the usual Euclidean distance between c and d. Thus, $h(C, D)$ is the smallest amount by which we need to "grow" the points of D in order that all of C is covered by the grown set. Using the directed Hausdorff distance as the matching criterion thus allows us to *find* the pattern in the background. (In contrast, a least squares fit would not produce this type of match since all background points, including those that do not correspond to any of the pattern points, would influence what is considered to be the optimal placement of the pattern.)

1.1 Previous Work

Point set pattern matching has been an important problem in machine vision for some time. A number of different general strategies

———————————————

1. The *undirected* Hausdorff distance is defined as $H(C, D) = \max\{h(C, D), h(D, C)\}$.

———————————————

- *M.T. Goodrich and M.W. Orletsky are with the Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218.*
  E-mail: {goodrich, orletsky}@cs.jhu.edu.
- *J.S.B. Mitchell is with the Department of Applied Mathematics and Statistics, State University of New York, Stony Brook, NY 11794-3600.*
  E-mail: jsbm@ams.sunysb.edu.

have been used to approach the problem. Four such strategies, along with their advantages and disadvantages, are outlined below.

The Cluster Approach. The clustering approach ([28], [29], [31], [34], [36], [38]) involves associating *confidence values* with locations in a discretized configuration space of possible orientations of the pattern with respect to the background and then choosing the match that is associated with the largest cluster or peak in the confidence values in the configuration space. These strategies are particularly effective at matching patterns that are partially occluded, or have points missing for other reasons, and patterns in which some points are severely corrupted, as demonstrated by the experimental work of the authors. The methods, however, require that the tolerance t be prespecified, and sometimes "tweaked," and produce matches in which those points that do not fall within t of an associated point are neglected in terms of the degree to which they actually deviate from the nearest matching point.

The Absolute Orientation Approach. The absolute orientation approach ([7], [20], [23], [24], [37]) is concerned with determining the pose (see, e.g., [16], [20]) of the pattern with respect to the background that minimizes the least squares error. These results assume that the size of the pattern set and the size of the background set are the same, and even more limiting, that a correspondence between the points in the pattern and the points in the background has already been established.

The Extracted Information Approach. Another general strategy, which we call the *extracted information approach*, attempts to match the pattern to the background based on information extracted from the sets of points. See, e.g., [2], [4], [17], [19], [32]. As shown by the authors, these methods work very well, theoretically and experimentally, for patterns and backgrounds that are related to each other by certain, sometimes strict, criteria. These methods do not, in general, work very well for the cases of missing points and, in some cases, the extracted information will change severely and abruptly with infinitesimal changes in a single pattern point.

The Computational Geometry Approach. Using computational geometry techniques, Alt et al. [5] give methods for finding *congruences* between two sets of points A and B under rigid motions. In addition to exact methods, they introduce an *approximate* version of the problem, for a given tolerance $\epsilon > 0$, and ask to find a motion T, if it exists, that allows a matching between each point in $T(A)$ and a point in B at distance $\leq \epsilon$. They also consider the *optimization* version, to compute the smallest ϵ admitting such a motion; unfortunately, their running times for this version are quite high. This version of the problem is very close to the problem we address in this paper.

Imai et al. [27] show that these bounds can be reduced somewhat if an assignment of points in A to points in B is given. Similarly, Arkin et al. [6] show that one can improve the running times in the approximate case if the "noise regions" are disjoint. Even so, the methods in these papers are relatively sophisticated, with rather high running times for all but the most simple motions.

In work more directly related to this paper, several researchers [9], [10], [25], [26] have studied methods for finding rigid motions that minimize either the directed or undirected Hausdorff distance between the two point sets. All of these methods are based on intersecting higher-degree curves and/or surfaces, which are then searched (sometimes parametrically [1], [11], [12], [13], [30]) to find a global minimum. This reliance upon intersection computations

TABLE 1
THE ASYMPTOTIC RUNNING TIMES FOR AN OPTIMAL MATCH
AND FOR OUR APPROXIMATELY OPTIMAL MATCH,
HAVING A WORST-CASE APPROXIMATION FACTOR GUARANTEE

Motion	Optimal Match	Our Method	Factor
$\mathcal{T}$ in $\mathbb{R}^2$	$nm^2\log n$ [25]	$nm\log n$	2
$\mathcal{T}\mathbb{R}$ in $\mathbb{R}^2$	$n^2m^3\log n$ [9]	$n^2m\log n$	4
$\mathcal{T}$ in $\mathbb{R}^3$	$n^3m^2\log^2 n$ [25]	$nm\log n$	$2 + \epsilon$
$\mathcal{T}$ in $\mathbb{R}^d$	–	$nm\log n$	$2 + \epsilon$
$\mathcal{R}$ in $\mathbb{R}^3$	–	$n^2m\log n$	$4 + \epsilon$
$\mathcal{T}\mathbb{R}$ in $\mathbb{R}^3$	–	$n^3m\log n$	$8 + \epsilon$

Transformations considered are $\mathcal{T}$ = translation and $\mathcal{R}$ = rotation, and their combination. Here, $0 < \epsilon < 1$ is a fixed constant. (The optimal match method for translation in $\mathbb{R}^3$ is for the undirected Hausdorff distance.)

leads to algorithms that are potentially numerically unstable, are conceptually complex, and have running times that are high for all but the most trivial motions. Indeed, Rucklidge [35] gives evidence that such methods *must* have high running times.

The high running times of these methods motivated Heffernan [21] and Heffernan and Schirra [22] to consider an approximate decision problem for approximate point set congruence (they did not study point set pattern matching). Their general framework, for a given parameter $\epsilon > 0$, is to solve the approximate set congruence problem [5], except that one is allowed to "give up" if one discovers that ϵ is "too close" to the optimal value ϵ^* (i.e., the Hausdorff distance between $T(A)$ and B). By allowing algorithms to be "lazy" in this way, they show that the running times can be significantly improved. Unfortunately, this approach can result in a large computing time that yields *no* approximation, with the time increasing substantially if one tries to get a yes-or-no answer for an ϵ close to ϵ^*. Thus, it is difficult to use their methods to approximate ϵ^*.

1.2 Our Results

In this paper, we present a very simple approach for approximate point set pattern matching under rigid motions, where one is given a pattern set P of m points in $\mathbb{R}^d$ and a background B of n points in $\mathbb{R}^d$ and asked to find a rigid motion T that minimizes $h(T(P), B)$. Our methods are based on a simple "pinning" strategy. They are fast, easy to implement, and numerically stable. Moreover, since they are defined for the (more general) directed Hausdorff measure, they are tolerant of noise in the background.

Our methods are not exact, however. Instead, in the spirit of approximation methods for other hard optimization problems (e.g., NP-hard problems [14]), we derive algorithms that are guaranteed to come close to the optimal value, ϵ^*. In particular, each of our methods gives a rigid motion T such that $h(T(P), B) \le \alpha\epsilon^*$, for some small constant $\alpha > 1$. (We note that a similar use of approximation algorithms is taken by Alt et al. [3] for the problem of polygon matching.) Our results are summarized in Table 1.

We justify the implementability of our methods through an empirical study of the running time and the quality of the match of our methods when run on various input instances. We compare the performance with that of a more conventional procedure based on a branch-and-bound search of a discretized configuration space. Our results show that, in practice, our methods are fast and produce rigid motions with good matches. We also give some heuristics that speed up the running time in practice, while not improving the worst-case running time.

2 OUR ALGORITHMS

The input to our algorithms is a set B of n points in $\mathbb{R}^d$ and a set P of m points in $\mathbb{R}^d$, where dimension d is considered constant.

2.1 Pure Translation in $\mathbb{R}^d$

In this subsection, we give an efficient method for finding a translation T' such that $h(T'(P), B)$ is at most $2h(T_{opt}(P), B)$, where T_{opt} is an *optimal* translation, i.e., one that minimizes $h(T(P), B)$, taken over all translations T.

Pick some point $p \in P$ as a "representative" for P. For each $b \in B$, define T_b to be the translation that takes p to b. Our method, then, is to find $\min_{b \in B}\{h(T_b(P), B)\}$ as our best approximate match, and let T' be the translation T_b that achieves this bound. This can clearly be done in $O(nm \cdot Nearest_d(n))$, where $Nearest_d(n)$ is the time needed to perform a nearest-neighbor query in an n-point set (in this case B) in $\mathbb{R}^d$.

Of course, if $d = 2$, then we can achieve $Nearest_2(n) = O(\log n)$ by answering nearest-neighbor queries using point location in a Voronoi diagram for B (e.g., [15], [33]), which requires $O(n \log n)$ preprocessing. For higher dimensions, this approach is not as efficient, however. So, our method for implementing nearest neighbors will instead be based on the (practical) method of Arya et al. [8], which finds *approximate* nearest neighbors in $O(\log n)$ time in $\mathbb{R}^d$, for any constant dimension $d \ge 2$, after $O(n \log n)$ preprocessing. Their method can be tuned to return a point whose distance is at most a $(1 + \epsilon)$-factor larger than the nearest-neighbor distance, for any constant $0 < \epsilon < 1$.

LEMMA 2.1. $h(T'(P), B) \le 2h(T_{opt}(P), B)$ in $\mathbb{R}^2$ and $h(T'(P), B) \le (2 + \epsilon)$ $h(T_{opt}(P), B)$ in $\mathbb{R}^d$, for $d \ge 3$ and for any constant $\epsilon \in (0, 1)$.

PROOF. (For $\mathbb{R}^2$) For simplicity of expression, define $h_{opt} = h(T_{opt}(P), B)$. Observe that, for each $p \in T_{opt}(P)$, there exists an associated $b \in B$ that is within a distance h_{opt} of p. Consider the process of translating the entire pattern $T_{opt}(P)$ so that a particular point p now coincides with its associated background point b. This translation will cover a distance of at most h_{opt} and will therefore increase the distance from any other point in the pattern to its associated background point by at most h_{opt}. Therefore, it will have a directed Hausdorff distance of at most twice that of $T_{opt}(P)$. This translation will be one of those generated and checked by our algorithm. Thus, our algorithm will produce a translation that results in a directed Hausdorff distance that is at most a factor of two times the minimal.

(For $\mathbb{R}^d$) For the case of $\mathbb{R}^d$, with $d \ge 3$, an identical argument would apply if we were to use an exact nearest-neighbor algorithm to compute the quality of the various translations considered. However, we are using the *approximate* nearest-neighbor algorithm of Arya et al. [8], so the observed directed Hausdorff distance of any translation $T(P)$ may appear to be greater (worse) than its actual value by a factor of up to $1 + \epsilon'$, where ϵ' is a parameter of the approximate nearest neighbor algorithm. Since one of the candidate translations will have a directed Hausdorff distance within a factor of 2 of the absolute optimal (by the argument above), our algorithm will select as the best translation one that has a directed Hausdorff distance no greater than $2(1 + \epsilon')h_{opt}$. Selecting $\epsilon' = \epsilon/2$ then gives the desired result. ☐

2.2 Translation and Rotation in $\mathbb{R}^2$

For points in the plane, we give an efficient method for finding a Euclidean motion (translation and rotation) E' such that $h(E'(P), B)$ is at most $4h(E_{opt}(P), B)$, where E_{opt} is an *optimal* Euclidean motion, i.e., one that minimizes $h(E(P), B)$, taken over all valid motions E.

Select from the pattern diametrically opposing points and call them r and k; this can be done trivially in time $O(m^2)$, but $O(m \log m)$ suffices [33]. Point r is treated as both the distinct representative of the pattern for the translation part of the transformation and it is treated as the center of rotation for the rotation part of the trans-

formation. Specifically, for each $b \in B$, define T_b to be the translation that takes r to b. Also, for a $b' \in B$, $b' \neq b$, define $R_{b'}$ to be the rotation about r that makes r, b', and k collinear. Let $E_{b,b'}$ be the Euclidean motion that is the combination of T_b and $R_{b'}$. Our method is to find $\min_{b,b' \in B}\{h(E_{b,b'}(P), B)\}$ as our best approximate match, and let E' be the Euclidean motion $E_{b,b'}$ that achieves this bound. This can be done in $O(n^2 m \cdot Nearest_3(n))$ time, which is $O(n^2 m \log^2 n)$ if one uses the best current point location method for a three-dimensional convex subdivision [18] to query nearest neighbors in a three-dimensional Voronoi diagram (e.g., see [15], [33]). Our preference, however, is to achieve a faster (and more practical) $O(n^2 m \log n)$ time bound using the approximate nearest neighbors method of Arya et al. [8], at a slight cost in the approximation factor.

LEMMA 2.2. $h(E'(P), B) \leq (4 + \epsilon)h(E_{opt}(P), B)$, for any constant $0 < \epsilon < 1$.

PROOF. Since the ϵ term is a direct consequence of our using approximate nearest neighbor searching to achieve $Nearest_3(n)$ $= O(\log n)$, it is sufficient to show that actual nearest neighbors would give an approximation factor of 4. For simplicity of expression, define $h_{opt} = h(E_{opt}(P), B)$. Observe that, for each $p \in E_{opt}(P)$, there exists an associated $b \in B$ that is within a distance h_{opt} of p. Consider the process of translating the entire pattern $E_{opt}(P)$ so that the particular point r now coincides with its associated background point b. This translation will cover a distance of at most h_{opt} and will therefore increase the distance from any other point in the pattern to its associated background point by at most h_{opt}. Now, consider the process of rotating the entire pattern about point r so that the line containing r and k now passes through the background point associated with k in $E_{opt}(P)$. This rotation will have the effect of moving point k by at most $2h_{opt}$. Since k is the furthest point in the pattern from the center of rotation, all other pattern points will be moved by a distance of at most $2h_{opt}$. Thus, any given point in the pattern can be moved by at most h_{opt} during the translation and at most $2h_{opt}$ during the rotation, and could have been initially at most h_{opt} away from its associated background point. Therefore, each point in the pattern will be at most a distance of $4h_{opt}$ from a background point. The pattern in its current position coincides with one of the Euclidean transformations generated and checked by our algorithm. □

2.3 Pure Rotation in $\mathbb{R}^3$

For points in $\mathbb{R}^3$, we give an efficient method for finding a (pure) rotation R', about the origin, such that $h(R'(P), B)$ is at most $4h(R_{opt}(P), B)$, where $R_{opt}(P)$ is an *optimal* rotation, i.e., one that minimizes $h(R(P), B)$, taken over all rotations R.

Find a point $p_1 \in P$ that is furthest from the origin. Find a point $p_2 \in P$ that has the maximum perpendicular distance to the line defined by the origin and point p_1. (It takes $O(m)$ time to find p_1 and p_2.) For each $b' \in B$, define $R1_{b'}$ to be the rotation that makes the origin, p_1, and b' collinear. For each $b'' \in B$, $b'' \neq b'$, define $R2_{b''}$ to be the rotation about the origin-p_1 axis that makes the origin, p_1, p_2, and b'' coplanar. Our method, then, is to find

$$\min_{b',b'' \in B}\left\{h\left(R2_{b''}\left(R1_{b'}(P)\right), B\right)\right\}$$

as our best approximate match, and let R' be the resultant rotation $R2_{b''}(R1_{b'}(P))$ that achieves this bound. This requires

$$O(n^2 m \cdot Nearest_3(n))$$

time. As above, we achieve $Nearest_3(n) = O(\log n)$ using approximate nearest-neighbor searching [8], and end up with the following result:

LEMMA 2.3. $h(R'(P), B) \leq (4 + \epsilon)h(R_{opt}(P), B)$, for any constant $0 < \epsilon < 1$.

PROOF. For simplicity of expression, define $h_{opt} = h(R_{opt}(P), B)$. Observe that, for each $p \in R_{opt}(P)$, there exists an associated $b \in B$ that is within a distance h_{opt} of p. Consider the process of rotating the entire pattern $R_{opt}(P)$ so that p_1, the furthest pattern point from the origin, now becomes collinear with its associated background point b and the origin. This process can move any point in the pattern by at most h_{opt}. Now, consider the second rotation (about the line through the origin and p_1) that brings p_2 coplanar with its matching background point. This rotation may move p_2 a distance of at most $2h_{opt}$ and, therefore, it may move any point in the pattern by at most $2h_{opt}$. These combined rotations move any pattern point at most a distance of $3h_{opt}$ from its original position, which is known to be within a distance of h_{opt} of a background point. Therefore, each point in the pattern will be a distance of at most $4h_{opt}$ away from a background point. This rotation will be one of those generated and checked by our algorithm. □

2.4 Translation and Rotation in $\mathbb{R}^3$

For points in $\mathbb{R}^3$, we give an efficient method for finding a Euclidean transformation E' such that $h(E'(P), B)$ is at most $(8 + \epsilon)h(E_{opt}(P), B)$, where E_{opt} is an *optimal* Euclidean transformation, i.e., one that minimizes $h(E(P), B)$, taken over all such transformations E.

Select from the pattern diametrically opposing points[2] and call them r and k. Choose a point $l \in P$ such that the perpendicular distance from l to the line rk is maximum. For each $b \in B$, define T_b to be the translation that takes pattern point r to b. For each $b' \in B$, $b' \neq b$, define $R1_{b'}$ to be the rotation that causes r, k, and b' to become collinear. For each $b'' \in B$, $b'' \neq b'$, $b'' \neq b$, define $R2_{b''}$ to be the rotation about the rk axis that brings b'' into the (r, k, l)-plane.

Our method, then, is to compute the value of

$$\min_{b,b',b'' \in B}\left\{h\left(R2_{b''}\left(R1_{b'}\left(T_b(P)\right)\right), B\right)\right\}$$

as our best approximate match, and let E' be the Euclidean transformation $R2_{b''}(R1_{b'}(T_b(P)))$ that achieves this bound. This can be done in $O(n^3 m \cdot Nearest_3(n))$ time. As above, we achieve $Nearest_3(n) = O(\log n)$ using approximate nearest-neighbor searching [8], and end up with the following result:

LEMMA 2.4. $h(E'(P), B) \leq (8 + \epsilon)h(E_{opt}(P), B)$, for any constant $0 < \epsilon < 1$.

PROOF. For simplicity of expression, define $h_{opt} = h(E_{opt}(P), B)$. In addition, as in previous proofs, we show that the expansion factor is 8 if one were to use actual nearest neighbors instead of approximate nearest neighbors. Observe that for each $p \in E_{opt}(P)$, there exists an associated $b \in B$ that is within a distance h_{opt} of p. Consider the process of translating the entire pattern $E_{opt}(P)$ so that r becomes coincident with its associated background point. This process can move any point in the pattern by a distance of at most h_{opt}. Now, consider the process of rotating the entire pattern so that line rk passes through the background point that is associated with k. This rotation can move any point in the pattern by at most $2h_{opt}$. Now, consider a second rotation that brings the background point associated with l into the (r, l, k)-plane. This rotation

---

2. While subquadratic algorithms exist for computing the diameter, we found it reasonable to use a simple $O(m^2)$ algorithm as a preprocessing step since we only need to perform this calculation once for the entire algorithm.

may move pattern point p_2 a distance of at most $4h_{opt}$ and, therefore, it may move any point in the pattern by at most $4h_{opt}$. This rotation will be one of those generated and checked by our algorithm. The translation may have moved any point a distance of at most h_{opt}, the first rotation may have moved any point a distance of at most $2h_{opt}$ farther, and the second rotation may have moved any point a distance of at most $4h_{opt}$ still farther. Considering that any given pattern point may have been a distance of h_{opt} away from its associated background point to start with, no pattern point can be farther than $8h_{opt}$ from its associated background point.   □

3 EXPERIMENTAL RESULTS

We have implemented our methods and conducted experiments comparing them with a method that produces best matches to an arbitrary precision using a conventional branch-and-bound search of a discretized configuration space. This conventional method seems to be the most practical previous best match procedure (we did not feel it was practically feasible to implement the previous intersection-based methods). As we show through our experimental results, however, this conventional method is still quite slow compared to our method, and the matches it finds are not that much better than the ones that our method finds.

Example Generation. The background points B are generated uniformly at random in the unit d-cube. We then randomly select m points from B to be an unperturbed pattern. We obtain a *perturbed pattern*, P, by perturbing each pattern point by a small amount (uniformly, in a ball of radius δ) so that the pattern no longer identically resembles a subset of the background points.

3.1 Implementation of the Approximate Match Algorithms

Pure Translation in $\mathbb{R}^d$. As described in Section 2.1, our approximate pattern matching algorithm translates the pattern so that the distinct representative of the pattern coincides with each of the n background points in succession. For each such translation, the directed Hausdorff distance is calculated and compared with the best found so far. If the new directed Hausdorff distance is smaller than the best found so far, the position of the pattern (i.e., the position of the distinct representative) and this new best distance replace those recorded so far. After the pattern has been translated to each of the background points, we output the best translation found (which is guaranteed to be within factor two of optimal).

There are various possible heuristics one can apply which do not improve the worst-case running time, but which do improve the running time in practice. We use a condition that terminates the while-loop early once it is known that a particular placement need not be further considered. Observe that in the calculation of the directed Hausdorff distance, we are finding the maximum amount by which a pattern point deviates from its nearest background point. As we determine this quantity for each of the pattern points we have a current maximum at any given point in the loop. If this current maximum ever exceeds the best directed Hausdorff distance found so far, the placement that we are checking is known to be suboptimal and does not warrant further consideration. We therefore terminate the while loop as soon the partial computation of the directed Hausdorff distance exceeds the global best found so far.

Translation and Rotation in $\mathbb{R}^2$. Diametrically opposing pattern points are chosen from the convex hull (in $O(m^2)$ time, as a preprocessing step), one of which will serve both as the distinct representative of the pattern and as the center of rotation. The algorithm then translates the pattern so that the distinct representative coincides with each of the n background points in succession. After each translation, the pattern is rotated about the current position of the distinct representative a total of $n-1$ times so that, after each rotation, the other antipodal point is aligned with another one of the background points. We now have the pattern in one of the $n(n-1)$ positions at which we check the directed Hausdorff distance. As with the translation-only case, we maintain the best directed Hausdorff distance found so far and the position of the pattern that produced it. If at any time one of the $n(n-1)$ placements has a directed Hausdorff distance that is better than the best found so far, our records are updated to reflect this new best position and directed Hausdorff distance. Again, we use an early loop-termination heuristic for speed.

Translation and Rotation in $\mathbb{R}^3$. We select the distinct representative and the antipode of the pattern, as we have done in $\mathbb{R}^2$ above. In this case, we also select a third pattern point, called the radial point, which has the property that it is the greatest distance away from the line passing through the distinct representative and the antipode. Our approximate match algorithm is comprised of three nested for-loops. The outer-most loop translates the pattern such that the distinct representative of the pattern coincides with each of the n background points in succession. The next loop chooses one of the remaining $n-1$ background points and rotates the pattern about the current position of the distinct representative so that the antipode becomes aligned with this selected background point. The inner-most for-loop selects a third background point from the remaining $n-2$ and performs a second rotation of the pattern, this time about the line passing through the current position of the distinct representative and the current position of the antipode, to bring the plane defined by the distinct representative, the antipode, and the radial point into a position that includes the background point chosen by this third for-loop. For each of the $n(n-1)(n-2)$ placements produced by the above described for-loops, the directed Hausdorff distance of the placement is generated and the current best is kept. At the termination of our algorithm, we output the best placement found.

3.2 Implementation of the Branch and Bound Algorithms

Pure Translation in $\mathbb{R}^2$. The conventional method against which we compared our method is a recursive algorithm. It receives a square defined by a center point and a side length. It then "probes" the center of the square by translating the pattern so that the distinct representative of the pattern is in the center of the square. For the pattern in this position, the directed Hausdorff distance is calculated. If this distance is the best found so far, it is recorded along with the probe point (center of square). The algorithm then recurses on each of the four quadrants. The recursion is terminated when it reaches a predefined maximum depth or if it is certain that placement of the distinct representative at any point in the square will not produce a directed Hausdorff distance that is better than the best found so far. One observation that we can use to terminate a branch of recursion early is that the directed Hausdorff distance can be decreased by an amount of at most x when the pattern is translated by a distance of x. If the value of the directed Hausdorff distance produced by probing the center of the square is so great relative to the best found so far that placing the distinct representative at any point in the square is known to produce a directed Hausdorff distance that does not beat the best found so far, we no longer need to search recursively this square and we can terminate this branch of the recursion.

Translation and Rotation in $\mathbb{R}^2$. The conventional method for translation and rotation in $\mathbb{R}^2$ involves searching the three-
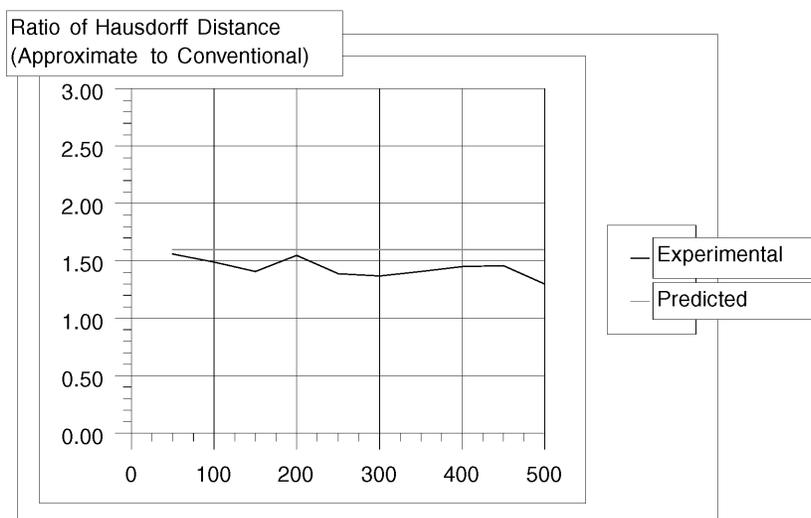
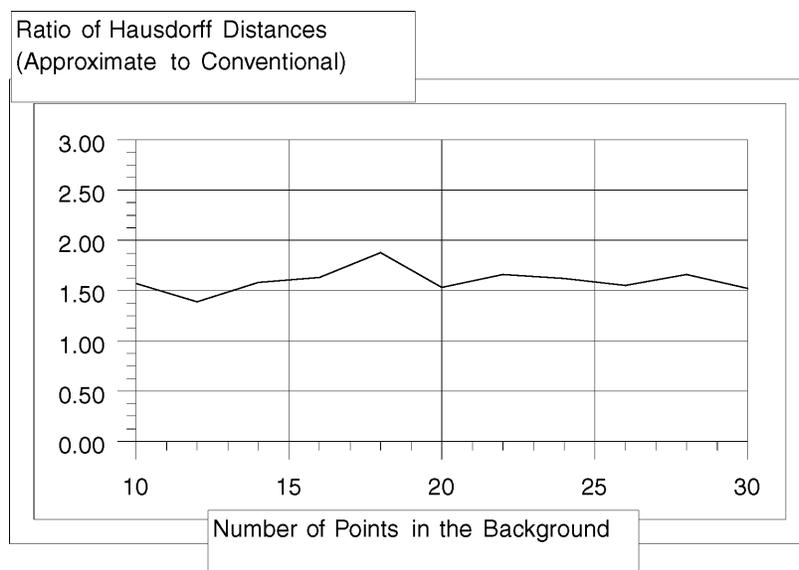Fig. 1. Ratio of the Hausdorff distances: Pure translation in $\mathbb{R}^2$.

Fig. 2. Ratio of the Hausdorff distances: Translation and rotation in $\mathbb{R}^2$.

dimensional configuration space in which the x and y positions of the distinct representative of the pattern comprise two of the dimensions, and the angular position, θ, of the pattern about the distinct representative comprises the third.

Translation and Rotation in $\mathbb{R}^3$. The conventional method for Translation and Rotation in $\mathbb{R}^3$ is again the search of a configuration space, which is now six-dimensional: three degrees of freedom (x, y, and z) in placing the distinct representative of the pattern and three rotational degrees of freedom (two in locating the antipode and one in orienting the pattern about the axis line through the antipode).

3.3 Experiment 1: Comparison of Match Qualities

While we have proven upper bounds on the worst-case behavior of our approximation algorithms, the goal of our first experiment is to see how close to optimal Hausdorff distance our method comes, in practice.

Pure Translation in $\mathbb{R}^2$. We have proven an upper bound of 2 on the ratio of the directed Hausdorff distance of our approximation to the directed Hausdorff distance of the *optimal* match un-

der translation. It is our conjecture that, for large sparse Bs and large sparse Ps, the approximate match algorithms will produce matches that are $(1 + \lambda)h_{opt}$, where λ is the ratio of the *expected* distance by which a point will be perturbed divided by the *maximum* distance by which a point will be perturbed; for our perturbation strategy in $\mathbb{R}^2$, this ratio will be

$$\lambda = \int_0^1 r \frac{2\pi\,dr}{\pi 1^2} = 2/3.$$

Our reasoning is as follows: If the pattern is large, it is likely that the absolute optimal placement of the pattern with respect to the background will be such that quite a few pattern points will be h_{opt} away from the nearest background point. The approximate-match algorithm produces, with high probability (especially, given the pattern-generation method used in these experiments), the match that is identical to this optimal match, differing only in that it is translated such that the distinct representative of the pattern is made to coincide with its associated background point. This translation will be in a direction that moves one or more of the poorly matching pattern points almost directly away from the associated background points.
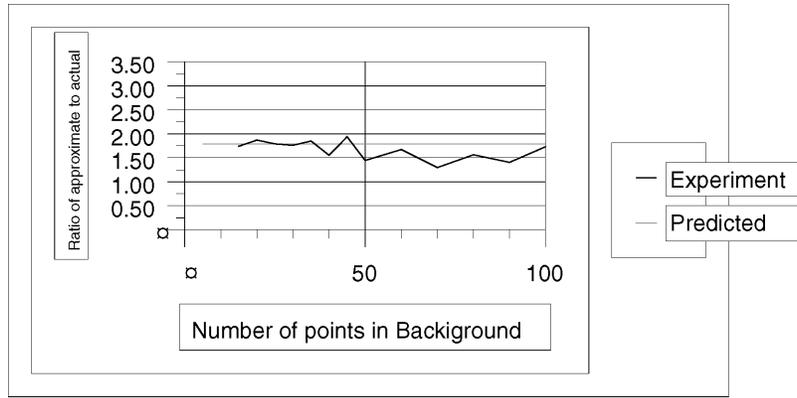
Fig. 3. Ratio of the Hausdorff distances: Pure translation in $\mathbb{R}^3$.
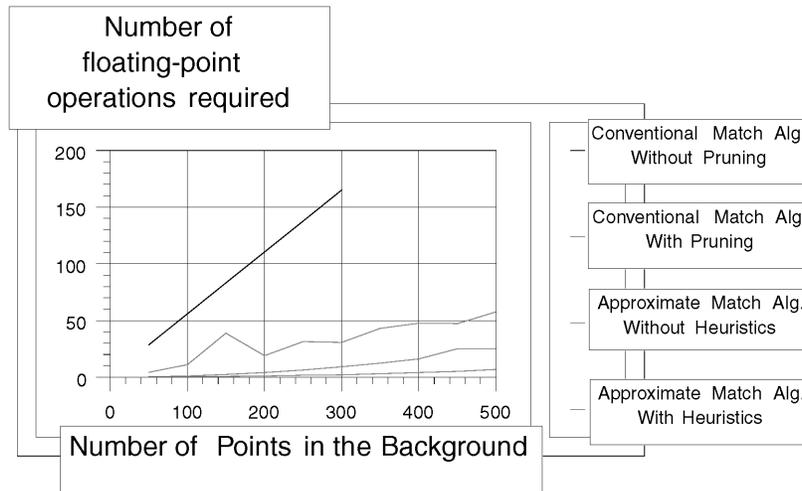
Fig. 4. Running times of the four methods (in Mega-FLOPs).

Thus, since $\lambda = \frac{2}{3}$, the expected Hausdorff distance for our algorithm will be $\frac{5}{3} \cdot h_{\text{opt}}$.

We conducted an experiment to test this hypothesis. One hundred sets of background points were generated, each having between 50 and 500 points. From each background, a pattern of size 10 was selected and perturbed. The pattern was then matched to its associated background using both the approximate match algorithm and a conventional match algorithm. The ratio of the directed Hausdorff distance of the match produced by the approximate match algorithm to the directed Hausdorff distance of the match produced by the conventional match algorithm is plotted in Fig. 1. The average of the ratios plotted is 1.44, which is close to the predicted value of 1.66. Note that the predicted value of this ratio assumes an infinitely large pattern and an unlimited depth of recursion in the conventional method. Decreasing either the pattern size or the depth of recursion would decrease the predicted value of the ratio and this too is reflected in this experiment. For this experiment, the conventional match algorithm was run to a depth of 11.

Translation and Rotation in $\mathbb{R}^2$. In Section 2.2, we have proven an upper bound of 4 on the approximation factor for our method. In order to determine the approximation factor observed in practice, 110 background point sets were generated, each having between 10 and 30 points. From each of these backgrounds, a 10 point pattern was selected and perturbed. The results of comparing the quality of the approximate match algorithm to the conventional branch-and-bound algorithm is plotted in

Fig. 2. The average ratio of the trials in this experiment is 1.60, which is substantially better than the worst-case ratio of 4.

Pure Translation in $\mathbb{R}^3$. For approximate matching under translation-only in $\mathbb{R}^3$, we can again expect match qualities that are in general better than the worst case bound. In a manner similar to that outlined for $\mathbb{R}^2$, the expected Hausdorff distance produced by the approximate match algorithm for matching under pure translation in $\mathbb{R}^3$ can be calculated: the expected Hausdorff distance for matches produced is $(1 + \lambda)h_{\text{opt}}$, where again λ is the expected distance by which a point is perturbed. For our perturbation strategy in $\mathbb{R}^3$,

$$\lambda = \int_0^1 r \frac{4\pi r^2 dr}{(4\pi/3)1^3} = 3/4 .$$

Our experiment used 45 sets of background points, each having between 5 and 100 points. From each background, a pattern of size 5 was selected and perturbed. The pattern was then matched to its associated background using both the approximate match algorithm and a conventional match algorithm. The ratio of the directed Hausdorff distance of the approximate match algorithm to the directed Hausdorff distance of the conventional match algorithm is plotted in Fig. 3. The average of the ratios plotted is 1.679, which is close to the predicted value of 1.75. The fact that the experimental average is less than the predicted value can be attributed to the relatively small pattern size of five points, which decreases the probability that one of the pattern points will be translated by a nearly maximal amount when the distinct representative of the pattern is translated to its associated background point.
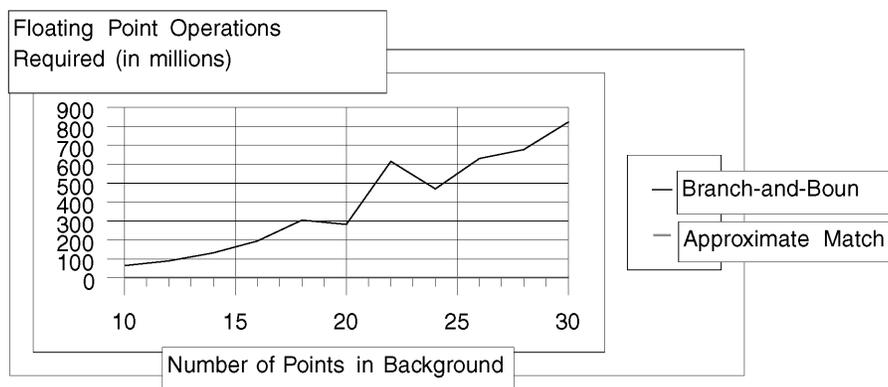
Fig. 5. Running times for matching under translation and rotation in the plane: Conventional method (Line 1) versus our approximate method (Line 2).
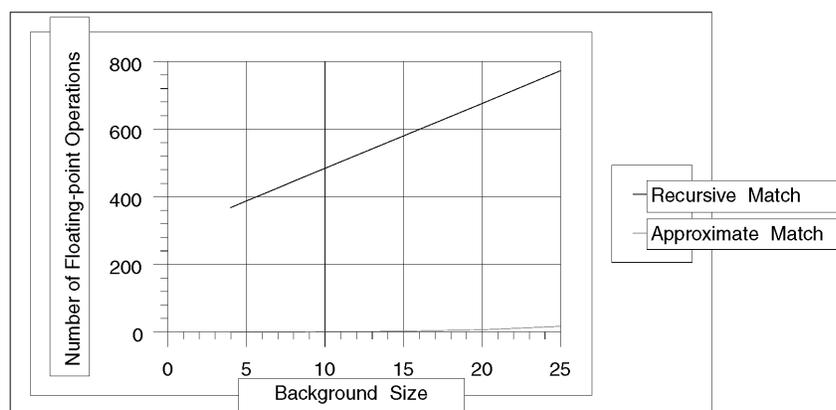

Fig. 6. Running times for matching under translation and rotation in $\mathbb{R}^3$: Conventional method (Line 1) versus our approximate method (Line 2).

3.4 Experiment 2: Running Times

The approximate match algorithms have worst-case time complexities that are much lower than those of the branch-and-bound match algorithms. We conjectured that the approximate match algorithms should also run faster in practice. Experiment 2 examines the extent to which they do, comparing them to what should be a good practical algorithm—branch-and-bound. To eliminate any system effects on the running time data, we keep a counter of floating point operations used by each algorithm.

Pure Translation in $\mathbb{R}^2$. In this case, 30 sets of background points were generated, each having between 50 and 500 points. From each background, a pattern of size 10 was selected and perturbed. The pattern was then matched to its associated background using both the approximate match algorithm and a conventional match algorithm, both with and without heuristic speedups. The results of this experiment are shown in Fig. 4. In every case, the approximate match algorithm, even without the heuristic speedup, had a smaller running time than the conventional algorithms.

Translation and Rotation in $\mathbb{R}^2$. In this case, 110 sets of background points were generated, each having between 10 and 30 points. From each background, a pattern of size 10 was selected and perturbed. The pattern was then matched to its associated background using both the approximate match algorithm and a conventional match algorithm (with heuristic speedups). The results of this experiment are shown in Fig. 5. In all 110 trials, the conventional match algorithm was slower than the approximate match algorithm by at least a factor of 442, the average slowdown being a factor of 1,199.

Translation and Rotation in $\mathbb{R}^3$. In this case, 24 sets of background points were generated, each having between 5 and 25 points. From each background, a pattern of size 5 was selected and perturbed. The pattern was then matched to its associated background using the approximate match algorithm with heuristic speedups, a depth-first branch-and-bound algorithm, and a breadth-first branch-and-bound algorithm. The running times of the depth-first and breadth-first branch-and-bound algorithms were, in all instances within a factor of 0.01 of each other and are, therefore, plotted as a single line in Fig. 6, which depicts the results of this experiment.

The branch-and-bound algorithms search a six-dimensional space comprised of three degrees of freedom in the translation of the pattern and three degrees of freedom in the rotation of the pattern. This produces a rather large branching factor of $2^6 = 64$ in the recursive algorithms and necessitated the depth of these algorithms to be limited to 3. With this (necessary) depth limitation, the approximate-match algorithm actually found better matches than the branch-and-bound algorithm did in all of the 24 cases, in spite of the fact that the branch-and-bound algorithms required on average 4,479 times as many floating-point operations. The ratio of the directed Hausdorff distance of the match produced by the depth-limited search to the directed Hausdorff distance of the match produced by the approximate match algorithm is plotted in Fig. 7. It should be noted that the largest of the breadth-first searches in this experiment consumed between one and two hours of real time on an otherwise unloaded Sun Sparc Station ELC running Sun OS 4.1.1.
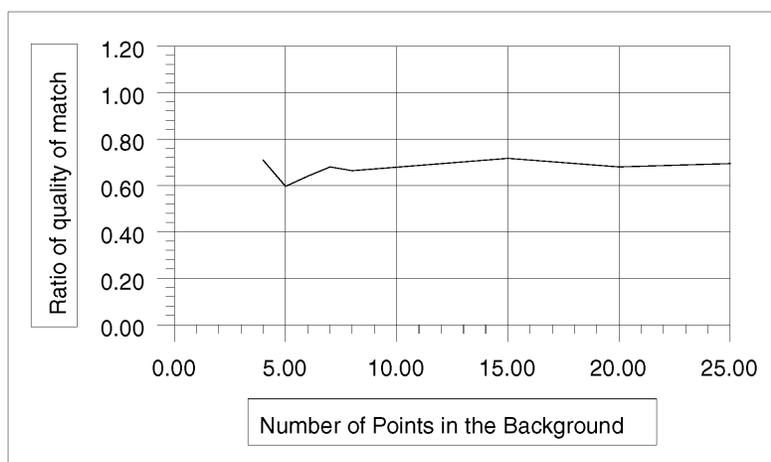
Fig. 7. The ratio of the directed Hausdorff distance of the match produced by the depth-limited search to the directed Hausdorff distance of the match produced by the approximate match algorithm for the case of translation and rotation in $\mathbb{R}^3$.
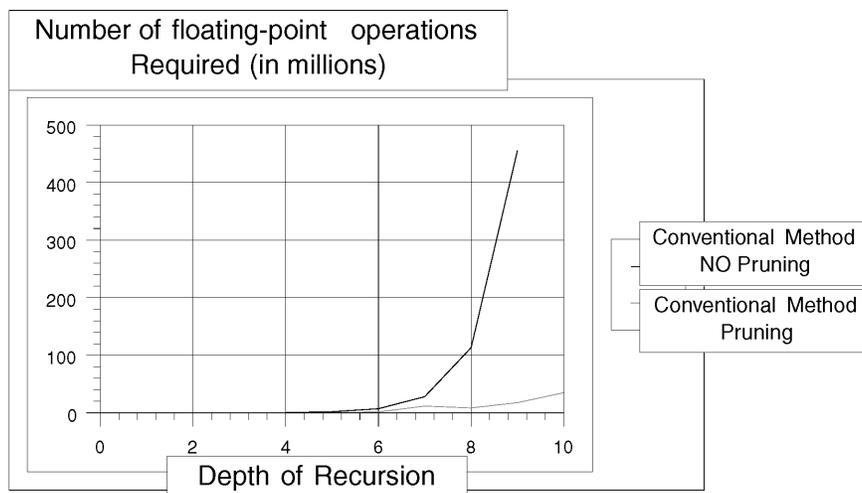

Fig. 8. Running times with and without pruning.

3.5 Experiment 3: Running Time versus Depth of Recursion

The depth of search of the conventional match algorithms that we have implemented must be limited. This experiment depicts the extent to which the running time of the algorithm increases as the depth of recursion is increased. Further, it shows the substantial speedup obtained by pruning the search. The results of this experiment are depicted in Fig. 8.

Four backgrounds were generated, each having 50 points. From each background, a pattern of size 10 was selected and perturbed. The conventional method with and without pruning was run ten times on each of the four data sets with the depth of recursion being varied from 1 to 10. An average of the running times of each of the four cases was taken and the results were plotted in Fig. 8.

4 DISCUSSION AND CONCLUSION

We have given approximate pattern matching algorithms for translation, rotation, and Euclidean transformations for point sets in two or more dimensions. Our algorithms are guaranteed to give a match with a directed Hausdorff distance that is no greater than a small constant times the best achievable directed Hausdorff distance. In addition, they have a time complexity that is substantially smaller than those of existing pattern

matching algorithms, they are easy to implement, and they run fast in practice.

Improving the Approximation Factors. We can extend our methods so that the match produced by the algorithm is arbitrarily close to optimal (within factor $(1 + \epsilon)$), while increasing the running time of the algorithm by only a constant factor (dependent on ϵ). We sketch the idea briefly for the case of matching under translation only in $\mathbb{R}^d$. When we pin the distinct representative of the pattern at a background point, we construct a d-dimensional uniform grid of points around that background point and pin the distinct representative at each of these grid points in succession. (The grid can be generated within a box whose size is given by the approximate Hausdorff distance computed using our unmodified approximation algorithm; the number of grid points depends on ϵ.) At each of these placements, we compute the Hausdorff distance and compare it against the best found so far. For cases such as Translation and Rotation in which placements are generated based on two or more background points (for example, a translation to place the distinct representative at one background point and then a rotation to align the antipode of the pattern with another background point, etc.), we can extend this idea using multiple grids in the obvious way.

REFERENCES

[1] P.K. Agarwal, B. Aronov, M. Sharir, and S. Suri, "Selecting Distances in the Plane," *Algorithmica*, vol. 9, pp. 495-514, 1993.

[2] N. Ahuja, "Dot Pattern Processing Using Voronoi Neighborhoods," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 4, no. 3, pp. 336-343, May 1982.

[3] H. Alt, B. Behrends, and J. Blömer, "Approximate Matching of Polygonal Shapes," *Annals of Math. and Artificial Intelligence*, vol. 13, pp. 251-266, 1995.

[4] H. Alt, L. Knipping, and G. Weber, "An Application of Point Pattern Matching in Astronautics," Technical Report B-93-16, Institut für Informatik, Fachbereich Mathematik und Informatic, Freie Universität Berlin, 1993.

[5] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl, "Congruence, Similarity and Symmetries of Geometric Objects," *Discrete Computing in Geometry*, vol. 3, pp. 237-256, 1988.

[6] E. M. Arkin, K. Kedem, J.S.B. Mitchell, J. Sprinzak, and M. Werman, "Matching Points into Pairwise-Disjoint Noise Regions: Combinatorial Bounds and Algorithms," *ORSA J. Computing*, vol. 4, no. 4, pp. 375-386, 1992.

[7] K.S. Arun, T.S. Huang, and S.D. Blostein, "Least Squares Fitting of Two 3-(D) Point Sets," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, pp. 698-700, 1987.

[8] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching," *Proc. Fifth ACM-SIAM Symp. Discrete Algorithms*, 1994.

[9] L.P. Chew, M.T. Goodrich, D.P. Huttenlocher, K. Kedem, J.M. Kleinberg, and D. Kravets, "Geometric Pattern Matching Under Euclidean Motion," *Computing Geometric Theory Applications*, vol. 7, pp. 113-124, 1997.

[10] L.P. Chew and K. Kedem, "Improvements on Geometric Pattern Matching Problems," *Proc. Third Scandinavian Workshop Algorithm Theory*, pp. 318-325. Springer-Verlag, 1992.

[11] R. Cole, "Slowing Down Sorting Networks to Obtain Faster Sorting Algorithms," *J. ACM*, vol. 34, pp. 200-208, 1987.

[12] R. Cole, "Parallel Merge Sort," *SIAM J. Computing*, vol. 17, no. 4, pp. 770-785, 1988.

[13] R. Cole, J. Salowe, W. Steiger, and E. Szemerédi, "An Optimal-Time Algorithm for Slope Selection," *SIAM J. Computing*, vol. 18, pp. 792-810, 1989.

[14] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. Cambridge, Mass.: MIT Press, 1990.

[15] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Heidelberg, West Germany: Springer-Verlag, 1987.

[16] D. Forsyth, J.L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell, "Invariant Descriptors for 3-D Object Recognition and Pose," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 971-991, Oct. 1991.

[17] J.W. Foster, G.K. Bennett, and P.M. Griffin, "Automated Visual Inspection: Quality Control Techniques for the Modern Manufacturing Environment," *Proc. 1987 IIE Integrated Systems Conf.*, pp. 135-140, Dec. 1987.

[18] M.T. Goodrich and R. Tamassia, "Dynamic Trees and Dynamic Point Location," *SIAM J. Computing*, vol. 28, no. 2, pp. 612-636, 1999.

[19] P.M. Griffin, J.W. Foster, and M.H. Han, "Automated Dimension Verification by Point Pattern Matching," *Proc. 1988 Int'l Industrial Eng. Conf.*, pp. 451-455, 1988.

[20] R.M. Haralick, C.N. Lee, X. Zhuang, V.G. Vaidya, and M.B. Kim, "Pose Estimation From Corresponding Point Data," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 19, no. 6, pp. 1,426-1,446, Nov.-Dec. 1989.

[21] P.J. Heffernan, "Generalized Approximate Algorithms for Point Set Congruence," *Proc. Third Workshop Algorithms Data Structure*, pp. 373-384, 1993.

[22] P.J. Heffernan and S. Schirra, "Approximate Decision Algorithms for Point Set Congruence," *Computing Geometric Theory Applications*, vol. 4, pp. 137-156, 1994.

[23] B.K.P. Horn, "Closed-Form Solution of Absolute Orientation Using Unit Quaternions," *J. Optical Soc. of Am. A*, vol. 4, no. 4, pp. 629-642, Apr. 1987.

[24] B.K.P. Horn, H.M. Hilden, and S. Negahdaripour," Closed-Form Solution of Absolute Orientation Using Orthonormal Matrices," *J. Optical Soc. of Am. A*, vol. 5, no. 7, pp. 1,127-1,135, July 1988.

[25] D.P. Huttenlocher, K. Kedem, and J.M. Kleinberg., "On Dynamic Voronoi Diagrams and the Minimum Hausdorff Distance for Point Sets Under Euclidean Motion in the Plane," *Proc. Eighth Ann. ACM Symp. Computer Geometry*, pp. 110-120, 1992.

[26] D.P. Huttenlocher, K. Kedem, and M. Sharir, "The Upper Envelope of Voronoi Surfaces and Its Applications," *Discrete Computing Geometry*, vol. 9, pp. 267-291, 1993.

[27] K. Imai, S. Sumino, and H. Imai, "Minimax Geometric Fitting of Two Corresponding Sets of Points," *Proc. Fifth Ann. ACM Symp. Computing Geometry*, pp. 266-275, 1989.

[28] D.J. Kahl, A. Rosenfeld, and A. Danker, "Some Experiments in Point Pattern Matching," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 10, no. 2, pp. 105-116, 1980.

[29] L.J. Kitchen, "Relaxation for Point-Pattern Matching: What It Really Computes," *Proc. IEEE Conf. Visualization and Pattern Recognition*, pp. 405-407, 1985.

[30] N. Megiddo, "Applying Parallel Computation Algorithms in the Design of Serial Algorithms," *J. ACM*, vol. 30, pp. 852-865, 1983.

[31] H. Ogawa, "Labeled Point Pattern Matching by Fuzzy Relaxation," *Pattern Recognition*, vol. 17, no. 5, pp. 569-573, 1984.

[32] H. Ogawa, "Labeled Point Pattern Matching by Delaunay Triangulation and Maximal Cliques," *Pattern Recognition*, vol. 19, no. 1, pp. 35-40, 1986.

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

[34] S. Ranade and A. Rosenfeld, "Point Pattern Matching by Relaxation," *Pattern Recognition*, vol. 12, pp. 269-275, 1980.

[35] W. Rucklidge, "Lower Bounds for the Complexity of the Hausdorff Distance," *Proc. Fifth Canadian Conf. Computer Geometry*, pp. 145-150, 1993.

[36] G. Stockman, S. Kopstein, and S. Benett, "Matching Images to Models for Registration and Object Detection via Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 4, no. 3, pp. 229-241, May 1982.

[37] S. Umeyama, "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376-380, Apr. 1991.

[38] C. Wang, H. Sun, S. Yada, and A. Rosenfeld, "Some Experiments in Relaxation Image Matching Using Corner Features," *Pattern Recognition*, vol. 16, no. 2, pp. 167-182, 1983.