Combinatorial and Experimental Results for Randomized Point Matching Algorithms

SANDY IRANI*   PRABHAKAR RAGHAVAN†

Abstract

The subject of this paper is the design and analysis of Monte Carlo algorithms for two basic matching techniques used in model-based recognition: alignment, and geometric hashing. We first give analyses of our Monte Carlo algorithms, showing that they are asymptotically faster than their deterministic counterparts while allowing failure probabilities that are provably very small. We then describe experimental results that bear out this speed-up, suggesting that randomization results in significant improvements in running time. Our theoretical analyses are not the best possible; as a step to remedying this we define a combinatorial measure of self-similarity for point sets, and give an example of its power.

1 Overview

Model-based recognition is an important component of computer vision. We study Monte Carlo algorithms for two basic approaches to model-based object recognition: *alignment*, and *geometric hashing*. We present analyses and experimental data for randomized versions of alignment and geometric hashing, studying the running times and failure probabilities. We further develop a combinatorial measure of self-similarity for point sets, and describe how this can lead to tighter analyses. Our (theoretical as well as experimental) results suggest that it is possible to avoid the worst case behavior of the deterministic versions of alignment and geometric hashing, while ensuring that the likelihood that a match is missed is very small. Most of the running times for model-based recognition reported in the literature are for *successful searches* in which the model *is* present in the scene. In fact, our analysis clearly highlights that the bad case is when the model is *not* present in the scene. Indeed, this is the case where randomization can help significantly in lowering running times, at the expense of a small chance that the algorithm declares that there is no match when in fact there is one.

The problem of model-based recognition is that of finding an occurrence of an object in a scene. Typically, features (such as points or line segments) are extracted from the images of the scene and model so that the problem becomes one of matching features. In this paper, we will assume that the features are points that have already been obtained from the image. Thus, the *scene* is a set S of n points in the plane, and the *model* is a set M of m points in the plane. A *transformation* is an affine mapping A that

*Dept. of Information and Computer Science, University of California, Irvine, CA 92717. Supported in part by NSF Grants CCR-9309456 and GER-94-50142.

†IBM Almaden Research Center, 650 Harry Road, San Jose CA 95120.

maps points in the plane to points in the plane; we allow transformations A that perform scaling, rotation, translation and reflection about a line. Note that for any two pairs of distinct points, (p_1, p_2) and (q_1, q_2), there are exactly two such transformations which simultaneously map p_1 to q_1 and p_2 to q_2 (each is a reflection of the other about the line through q_1 and q_2). All our theoretical results generalize in obvious ways to broader classes of transformations and to higher dimensions, but we do not detail these here.

For a point p and a transformation A, we denote by p_A the point obtained by applying A to p. Similarly, for a set of points Q, we denote by Q_A the set of points that results from applying A to the points of Q. For a positive constant $\alpha \in [0,1]$, we say that a scene S contains an *α-match* for the model M if there is a transformation A that maps at least a fraction α of the points of M into a subset of S; symbolically, $\exists A, M'$ such that $M'_A \subseteq S$, where $M' \subseteq M$ and $|M'| \geq \alpha|M| = \alpha m$. We say that the transformation A is an *α-matching*. Here α is a threshold for declaring a match to be successful; $\alpha = 1$ demands a perfect match in which all points of M (after transformation by A) are found in S. A value $\alpha < 1$ is more likely to be used in practice to allow for part of the model to be occluded in the scene and the effects of noise.

When checking whether a point p belongs to a scene S, we declare that $p \in S$ if there exists a point q in S that is within distance ϵ of p for a small fixed positive constant ϵ. This is to cope with the effects of discretization in generating the model as well as in the process of transformation. In light of this, $M_A \cap S$ is the set of points $p \in M_A$ such that there is a point in S that is within distance ϵ of p. The minimum ϵ such that there is a match of M into S is called the one-way *Hausdorff distance*. There is an extensive body of work on computing the minimum Hausdorff distance (see [CGHKKK] and the references therein). We do not seek to minimize this ϵ but rather adopt an acceptable threshold for ϵ that is small compared to the inter-point distances in the scene.

We pause to dwell briefly on the important issue of errors that is highlighted in the computer vision literature — errors due to noise in measurement when extracting points from an image, errors due to finite-precision computation, etc. How such errors affect the performance of recognition algorithms has been the subject of several investigations [GH90, LW91]. These studies have focused on the question of whether geometric hashing in particular amplifies noise in the data more than other methods. The important point here is that although the performance of each method will degenerate differently in the presence of more noise in the data, such errors affect the randomized algorithms we propose as well as their deterministic antecedents, equally. This will be clear from the fact that our randomized algorithms are exactly the same as the original deterministic algorithms, augmented by some random sampling steps.

The only additional error stemming from randomization is the *one-sided error* common in Monte Carlo algorithms: if the algorithm declares that there is an α-match, then it is error-free and can exhibit an α-matching transformation. On the other hand, there is a small probability that the algorithm declares that there is no α-match when one exists. Our analysis will bound this failure probability.

Section 2 gives basic analyses for randomized alignment and geometric hashing. Section 3 gives the results of our experiments that show the speedups yielded by randomization. In Section 4 we give an approach to improving the analysis of Section 2 by applying a measure of self-similarity we define. This allows us to recast our analysis in terms of the self-similarity of the model/scene. Such reasoning suggests that the worst case for the alignment method is when the model or scene has many regularly spaced points (as in a lattice). We then use our refined analysis to show in Corollary 11 that when the model or scene is a grid one can improve on the analysis of Section 2. This leads to the main open question stemming from this work: to improve on the analysis for the general case, one would either have to prove formally that the worst case occurs when the scene or model is a grid, or that our bounds on the self-similarity measure can be extended to general point sets.

2 Basic Analyses

2.1 The Alignment Method

In the deterministic version of the alignment method [HU87, HU90] an ordered pair of points is chosen from the model; for every ordered pair of distinct points in the scene, the two transformations A_1 and A_2 that map the model pair to the scene pair are computed. Each transformation A thus generated is tested with the entire model to see if $|M_A \cap S| = m$. In a more realistic scenario, some model points will be occluded in the scene. Let α be a fixed positive constant. The alignment method will only find an α-match if the pair of model points occurs in an α-match in the scene. We call a pair of points (p, p') in the model a *matching pair for* A if $|M_A \cap S| \geq \alpha m$, $p_A \in (M_A \cap S)$, and $p'_A \in (M_A \cap S)$. For a given transformation A which produces a match, we are only guaranteed that there are $(\alpha m)(\alpha m - 1)$ matching pairs for A out of the $m(m-1)$ pairs of points in the model. Thus, deterministically searching for a matching pair for A could require examining $\Theta(m^2)$ pairs in the worst case. Since for each such pair, we spend $\Theta(n^2 m \log n)$ time searching for a match, the alignment method has a worst case running time of $\Theta(m^3 n^2 \log n)$. Our first use of randomization is in choosing the pair of points from the model. A note on random sampling: in general one may sample from a set with or without replacement. Sampling with replacement leads to considerably simpler expressions in the analyses. However, the difference in the probabilities of events of interest is negligible in the two cases (see for instance the treatise by Feller [Fel50] on the effects of sampling with and without replacement). In practice, sampling without replacement is slightly preferable. Thus, in all our experiments, we use sampling without replacement. In all our analyses, we use sampling with replacement, noting that this is only for simplicity and that in the limit, the probabilities of the events we study is the same as without replacement.

Proposition 1 *If A is α-matching, then a randomly chosen pair of points from the model fails to be a matching pair for A with probability at most* $(1 - \alpha^2)$.

Proof Since $|M_A \cap S| \geq \alpha m$, then for a randomly chosen point p from M, $p_A \in S$ with probability at least α. The probability that for two randomly chosen points p and q, either $p_A \notin S$ or $q_A \notin S$, is at most $1 - \alpha^2$. ∎

Thus, the expected number of trials before we find a matching pair is a constant, circumventing the worst case of $\Omega(m^2)$ for the deterministic algorithm. This already gives us a factor of m^2 improvement in running time. (A technicality: the analysis assumes that scaling is not so extreme that most matching pairs are transformed into a single point of the scene, a reasonable assumption in practice. Theoretically though, it is possible that almost all the points are concentrated in a very small region; even in this anomalous case randomization yields a factor of m improvement in running time.) This approach is also employed by Fischler and Bolles [FB81]. However in their problem, they already know the correct correspondence of model points to scene points, whereas our focus is on *finding* this correspondence.

Our second use of randomization is motivated by the observation that after a matching pair has been found, an additional running time of $\Omega(mn^2 \log n)$ in the wirst case for the deterministic algorithm is not a theoretical artifact, but actually occurs whenever the model is not present in the scene. Suppose we are given a pair of points in the model. There are $2n(n-1)$ transformations obtained from mapping the model pair to every pair of points in the scene. It takes $\Theta(m \log n)$ time to determine whether any one of these transformations is α-matching.

To avoid this bottleneck, instead of testing the entire set of points in the model on every transformation, we pick a random subset R of r model points and use these points to test the $2n^2$ transformations. If for some transformation A, λr of the points in R match (where λ is a judiciously chosen constant — see below), we test the entire model with A to see if we have an α-match. If l is the number of times that λr points in R match, the running time for the algorithm is $\Theta((n^2 r + lm) \log n)$. Even for small r, l is generally very small in practice. Note that even though the worst case running time of the randomized

Algorithm RandomAlign(λ, α, r):

1. Select a random subset R of r points from M.
2. Select an ordered pair (p_1, p_2) at random from R.
3. For each ordered pair (q_1, q_2) in the scene:
   (a) Find A_1 and A_2 which map (p_1, p_2) to (q_1, q_2).
   (b) For each $i \in \{1, 2\}$,
   If ($|R_{A_i} \cap S| \geq \lambda r$) Test if $|M_{A_i} \cap S| \geq \alpha m$. If so, STOP.

Figure 1: *The alignment method using random sampling.*

scheme is $\Omega(n^2 m \log n)$, on typical inputs it is much smaller, whereas the deterministic scheme achieves the worst case whenever the model does not appear in the scene. In Section 4 we formalize this notion that "l is typically small" by introducing a measure of self-similarity for a point set and showing that it is in fact small for some cases.

When r is small, the running time tends to be faster as is indicated in our experimental results. However, this also increases the probability that an α-match is missed. Making no assumptions about the input points, we analyze the probability of such a miss as a function of α, r and λ. We call a subset R of r points from the model λ-*sufficient* for a transformation A if $|R_A \cap S| \geq \lambda r$.

Lemma 2 *Let A be an α-matching transformation and let R be a subset of r points chosen at random with replacement from M. The probability that R is not λ-sufficient for A is $\leq e^{-2(\lambda-\alpha)^2 r}$.*

Proof Let M' be the set of points in the model that match when A is applied. That is, $M' \subseteq M$ and $M'_A \cap S = M_A \cap S$. R fails to be λ-sufficient for A only if $|R \cap M'| < \lambda r$. Since we are sampling with replacement (see discussion above), as each point in R is drawn, the probability that it is in M' is $|M'|/|M| \geq \alpha$. Furthermore, the probability that a randomly chosen point from M is in M' is independent of the probability that any other randomly chosen point from M is in M'. Thus, if S_r is the number of points in M' out of r randomly chosen points,

$$Pr[S_r \leq \lambda r] = \sum_{i=1}^{\lambda r} \binom{r}{i} (1-\alpha)^{r-i} \alpha^i.$$

The expected size of $M' \cap R$ is αr. By a well known bound on the tail of the binomial distribution ([AS92], [Che52]), we can upper bound the probability that $|M' \cap R| \leq \lambda r$:

$$Pr[S_r \leq \lambda r] = e^{-2(\alpha-\lambda)^2 r}.$$

■

We can reduce the failure probability by repeating the procedure. he resulting algorithm RandomAlign(λ, α, r) employs randomization both in picking the initial pair of model points and in picking a random subset of model points. See Figure 1 for a formal description of the algorithm.

Theorem 3 *If there is an α-match of the model in the scene, then the probability that t iterations of* RandomAlign(λ, α, r) *fail to find one is at most $(1 - \alpha^2 + e^{-2(\lambda-\alpha)^2 r})^t$.*

Proof Let A be a transformation which produces an α-match. The theorem follows from Proposition 1 and Lemma 2 and the observation that in one iteration, the algorithm fails to find an α-match only if the random pair chosen from the model is not a matching pair for A or the random subset chosen from the model is not λ-sufficient for A. ■

To better appreciate the bound of Theorem 3, the reader may wish to consider the values $\alpha = 0.9$, $\lambda = 0.6$ and $r = 6$. Then, the bound of the theorem guarantees that the chance of failing to find an α-match when one exists is at most .006 for $t = 8$; further, the bound drops exponentially with t.

2.2 Geometric Hashing

Geometric hashing, developed in [LW88, LSW88, Wol90], divides the model recognition process into a *preprocessing* phase and a *query* phase. The preprocessing stage is done on the model, independently of the scene, and can be used to locate M in any scene. Preprocessing requires time $\Theta(m^3 \log m)$ during which a table of size $\Theta(m^3)$ is built. The table contains $2m^2$ transformations and $2m^3$ points and is constructed as follows. A pair of points (b_1, b_2) is chosen arbitrarily from the model. Then for every pair (p_1, p_2) of points in the model, we compute the two transformations A_1 and A_2 that simultaneously map p_1 to b_1 and p_2 to b_2. These transformations are stored in the table. Then for each $i \in \{1, 2\}$, and each point in $p \in M_{A_i}$, p is stored in the table of points with the tag 'A_i'. The points are indexed by their location in the plane. We make the assumption, as appears to be customary in the analysis of geometric hashing [LW88], that only a constant number of points map to the same location in the table. A typical point set is very likely to have this property.

In the query phase, the table is used to locate an α-match. For every pair of points (q_1, q_2) in the scene, we determine the two transformations that map q_1 to b_1 and q_2 to b_2. We call the resulting set of $2n^2$ transformations *scene*-transformations. For each such transformation A generated and for every point $p \in S$, we test if p_A is in the table, adding one vote for each transformation whose label appears in the entry for p_A. If there is a transformation in the table which receives at least αm votes, an α-match has been found. We can reconfirm the match by computing the transformation that maps the model to the scene directly and check that at least αm points in the model match under this transformation. Lookup in the table takes time $O(\log n)$; thus the total running time is $O(n^3 \log n)$.

The first use of randomization is similar to our use of randomization in the alignment method: we build the table using a random subset R of r points from the model. This reduces preprocessing time to $\Theta(r^3 \log r)$ and the storage to $\Theta(r^3)$. If in testing a scene-transformation A there is a transformation A' in the hash table that receives λr votes, we compute the transformation $B = (A^{-1} \circ A')$ that maps model points directly to scene points. If A is an α-matching, then the match will be found as long as R is λ-sufficient for A. If l is the number of potential transformations for which we check the entire set of model points, the running time is $\Theta((n^3 + lm) \log n)$. In the worst case, l could be as large as $\Omega(m^2 n^2)$, but typically the $O(n^3 \log n)$ term dominates even when r and λ are small.

The second use of randomization in geometric hashing is based on the observation that in order to find an α-matching transformation A, it is enough to test the scene-mappings obtained from mapping (q_1, q_2) to (b_1, b_2) where q_1 and q_2 are any two points in $R_A \cap S$. We will assume at this point that the randomly chosen subset R is λ-*sufficient* for A. We say that such a pair (q_1, q_2) is a *matching pair for* A and R. The probability that two randomly chosen points from the scene are a matching pair for A and R is at least $(\lambda r/n)^2$. The expected number of trials necessary to find a matching pair for A is $(n/\lambda r)^2$. For each pair chosen, it takes $\Theta(n \log n)$ time to test the two transformations generated by mapping the scene pair to the model pair. Thus, if we use $t(n/\lambda r)^2$ random pairs, and if l is the number of potential transformations for which we check the entire set of model points, the running time is $\Theta((tn^3/(\lambda r)^2 + lm) \log n)$. Since on almost all inputs, the $\Theta(tn^3 \log n/(\lambda r)^2)$ term dominates, the query time is shorter for larger r. Thus, r should be chosen to be as large as possible given any a priori constraints on storage and preprocessing time.

Algorithm BUILDTABLE(r):

1. Select a random subset R of r points from M.
2. For every ordered pair (p_1, p_2) in R:
   (a) Find A_1 and A_2 which map (p_1, p_2) to (b_1, b_2).
   (b) For each $i \in \{1, 2\}$ and For each $p \in R$,
   Store tag 'A_i' under entry p_{A_i}.

Figure 2: *Algorithm to build the hash table.*

Lemma 4 *Let A be an α-matching transformation. Let R be a subset which is λ-sufficient for A. If $t(n/\lambda t)^2$ pairs are picked at random from the scene, we will fail to find a matching pair for A with probability at most e^{-t}.*

Proof Let $x = (n/\lambda r)^2$. The probability that one randomly chosen pair is not a matching pair for A is at most $(1 - 1/x)$. The probability that all tx randomly chosen pairs are not matching pairs for A is at most

$$\left(1 - \frac{1}{x}\right)^{xt} \leq e^{-t}.$$

■

For $t = \log n$ the failure probability can be reduced to $1/n$ while obtaining a speedup of a factor of $((\alpha m)^2/\log n)$. We denote by GEOMETRICHASH(t, α, λ, r) the randomized algorithm which uses both randomized techniques and tries $t(n/\lambda r)^2$ pairs of points chosen randomly from the scene. See Figures 2 and 3.

Algorithm GEOMETRICHASH(t, λ, α, r):
Repeat $t(n/\lambda r)^2$ times:

1. Pick an ordered pair (p_1, p_2) at random from the scene.
2. Find A_1 and A_2 which map (p_1, p_2) to (b_1, b_2).
3. For each $i \in \{1, 2\}$,
   (a) For each $p \in S$:
      i. Search for p_{A_i} in table.
      ii. Add one vote for each tag 'A' under p_{A_i}.
   (b) For each A with $\geq \lambda r$ votes,
   Test if $|M_{A_i^{-1} \circ A} \cap S| \geq \alpha m$. If so, STOP.

Figure 3: *Algorithm to find model in scene using randomized geometric hashing.*

Theorem 5 *If there is an α-match of the model in the scene, then the probability that t' iterations of* GEOMETRICHASH(t, λ, α, r) *fails to find one is at most* $(e^{-t} + e^{2(\lambda-\alpha)r})^{t'}$.

Proof Let A be a transformation which produces an α-match. The theorem follows from Lemmas 2 and 4 and the observation that in one iteration, the algorithm fails an α-match only if the random subset chosen from the model is not λ-sufficient for A or every scene pair tested is not a matching pair for A and R. ∎

Notice that for smaller r, the preprocessing time is faster and the size of the table is smaller but the number of trials required to find a matching pair with some fixed probability is higher. If GEOMETRICHASH(t, λ, α, r) is iterated t' times, we keep t' tables, each requiring $O(r^3)$ in storage and $O(r^3 \log r)$ preprocessing time.

3 Experiments

We describe here a representative sample of our experimental results. In our experiments, we use three models: Model 1 is a screwdriver with 20 feature points, Model 2 is an unstapler with 20 feature points and Model 3 is a pair of scissors with 40 feature points. The scene is generated by applying transformations to the models and deleting some of the points (to simulate occlusion). The scene used for the experiments described in this section is represented by the table in Figure 4. Each number in the table represents an occurrence of the corresponding model in the scene. The number indicates the number of points occluded. Thus, there are no copies of the screwdriver, three copies of the unstapler and one copy of the scissors.

Occluded Points		
Model 1 Screwdriver	Model 2 Unstapler	Model 3 Scissors
-	0,2,2	4

Figure 4: *Number of points occluded from each copy of the models in the scene.*

For all of our experiments, we use $\alpha = .8$ as the match threshold. We vary r, the number of points in the random subset and λ, the threshold for the random subset. Every data point is the average of 20 runs. In each run, RANDOMALIGN is iterated exactly once in determining whether or not there is a match. We terminate the search if an α-match is found.

For each model/scene pair, we search for the model in the scene using the alignment method for every value of λ in $\{.4, .6, .8\}$ and every value of r in $\{6, 9, 12, 15\}$.

Then for each model/scene pair, we search for the model in the scene using geometric hashing for every value of r in $\{6, 9, 12, 15\}$. We test $(n/\lambda r)^2$ pairs of points chosen at random from the scene and run GEOMETRICHASH(t, λ, α, r) exactly once.

The running time of the deterministic algorithms is highly dependent on the order of the points in the scene. In order to smooth out this effect, we randomly permuted the points in the scene between trials of the deterministic algorithms. Thus, our reported runtimes for the deterministic algorithms do not penalize them as the theoretical bounds would; yet, as our data shows, our randomized algorithms perform significantly better.

Note that it was not our goal to fine-tune the algorithms for real-time performance. Instead we set up a uniform set of routines that permit a comparison between the randomized and deterministic methods, while re-using many common functions. The significant data is the relative values (rather than absolute magnitudes) of the runtimes. All experiments were run on an IBM RS/6000 PowerPC 25E.

3.1 Results and Interpretation

In all of our results, running times depended very little on the value of λ. This indicates that the false matches seldom result in the whole model being checked. We therefore only display running times in Figures 5–8 for $\lambda = .8$.

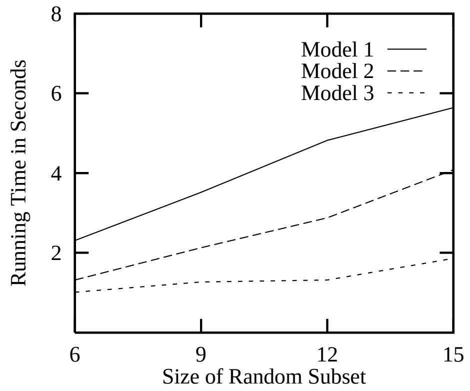

Figure 5: *Running times for* RandomAlign. *Deterministic running times — Model 1: 3516 seconds, Model 2: 2.04 seconds, Model 3: 10.33 seconds.*

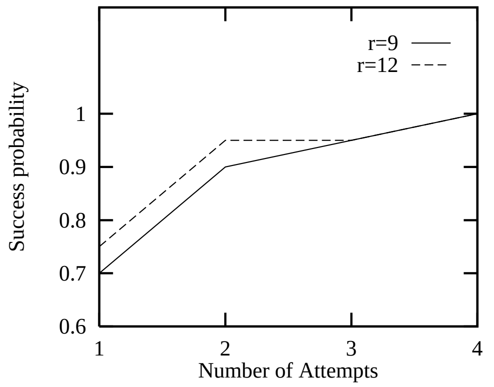

Figure 6: *Success probabilities for* RandomAlign *on Model 3.*

Figure 5 shows the running times of RandomAlign in searching Scene 1 with different values of r. The fastest running times occur when searching for Model 2. This is because there are three occurrences of Model 2 in the scene. Furthermore, for one of the occurrences, no points are occluded. The next fastest running times occur with Model 3 where there is one occurrence of the model with 10% occlusion. The slowest running times occur with Model 1 which does not appear in the scene.

As expected, the running time increases almost linearly with r, indicating that even for small r little time is spent checking potential matches with the entire set of points in the model. The caption indicates the running time of the deterministic algorithm. The large running time of the deterministic method comes from the fact that in the worst case, $(\alpha m)^2$ pairs from the model must be used with every pair of scene points to generate a potential transformation before the search is concluded. When the model is in the scene, even the deterministic search typically ends much earlier because an occurrence of the model is found. However, when the model is not in the scene, this results in much larger running times because all $(\alpha m)^2$ pairs are checked.

Model 3 is present in the scene with significant occlusion, a situation where the one-sided error of RandomAlign starts to become significant: in Figure 6 we depict the probability that RandomAlign succeeds in finding Model 3 in the scene for $r = 9$ and $r = 12$, as we vary the number of times (1 to 4) that it picks a random subset of the model and checks for a match. (For comparison, the results of Figure 5 correspond to a single random subset.) The results suggest that two random subsets give very good success probabilities; the runtimes for $r = 9$ and $r = 12$ are 2.78 seconds and 3.17 seconds, respectively. The running time is thus not substantially more than for a single random subset; if we go with two iterations routinely, the runtime degradation is worst when the model does not occur in the scene (where the observed runtime doubles, as expected).

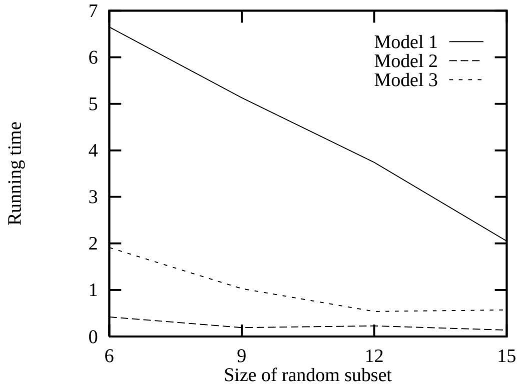

Figure 7: *Search times for* GeometricHash. *Deterministic search times — Model 1: 116.56 seconds, Model 2: 0.98 seconds, Model 3: 4.56 seconds.*

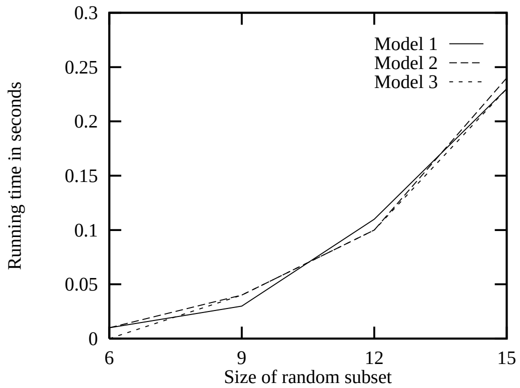

Figure 8: *Times to build hash tables in* GeometricHash. *Deterministic build times — Model 1: 0.63 seconds, Model 2: 0.64 seconds, Model 3: 8.22 seconds.*

Figures 7 and 8, respectively, show search and table-building times for GeometricHash with different values of r. As expected, the search time decreases as the size of the random subset used to build the table increases. Thus, it is desirable to keep r as large as possible subject to memory constraints. In addition, search times depended on the number of occurrences of the model in the scene and the amount of occlusion.

We also studied the variations of these times with the number of random pairs of scene points attempted. The expected number of pairs that must be tested before a match is found (if a match is indeed there) is at most $(n/\lambda r)^2$. We tested $t(n/\lambda r)^2$ pairs for different values of t. As depicted in Figure 9, the search times increased linearly with the number of pairs when the model was not present in the scene and increased only slightly when it was present. Again, the error probability in the presence of significant occlusion (Model 3) dropped as the number of scene pairs tested increased. (See Figure 10.)

Recall that there are two potential sources of failure in RandomAlign: the first is that the random subset chosen may not have λr matching points, and the second is that the random pair chosen from the

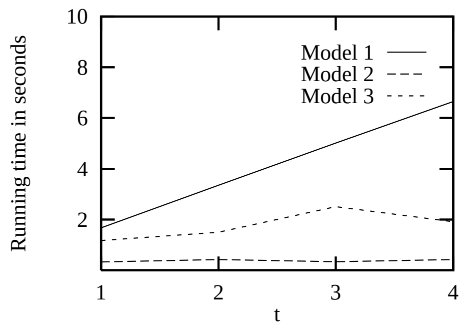

Figure 9: *Running time of Geometric Hashing as more random pairs from the scene are tested.* $r = 6$ *in all experiments.*

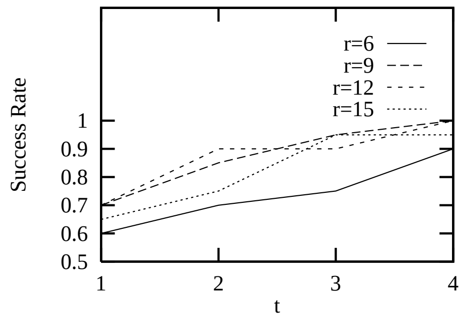

Figure 10: *Success probability of Geometric Hashing as more random pairs from the scene are tested.*

model may not be a matching pair. The fact that we could virtually eliminate the error for hashing by testing enough scene pairs and keeping the values of r and λ the same as in alignment, suggested that the source of the failure in alignment is in not picking a matching pair. Recall that Geometric hashing also requires picking a random subset with λr matching points. In practice, geometric hashing may not enjoy the same rate of success due to the effects of noise. However, the deterministic and randomized versions of geometric hashing will suffer equally in the presence of noise.

4 The self-similarity of point sets

In this section, we investigate ways to improve the worst-case bounds for point matching. In particular, we noted in Section 2 that the alignnment method obtains a bound of $\Theta((n^2r + lm)\log n)$, where l is the number of transformation which cause the randomly chosen subset of r points to match the scene. We observed that in our experiments, l is always very small. This naturally leads to the question of whether it is possible to choose a subset of points from any model such that l will be provably small for any scene. We note that in this section, we require points to coincide perfectly in order for them to constitute a match. This corresponds to requiring $\epsilon = 0$, however these results would also apply when the "fuzz-factor" ϵ is small with respect to the inter-point distances. Below, we analyze the problem of finding a perfect match of M in S where $\alpha = 1$. All our results can be generalized for $\alpha < 1$, while only effecting the running time by a constant factor.

Suppose that we had a subset R of the model with r points such that the number of times that R

matches perfectly into S is $n^2/f(n,r)$. Then we can use R to screen potential transformations and get a running time of $O(n^2 r + n^2 m/f(n,r))$. It is known that if r is taken to be a constant, then for any set R, there is an S such that R matches in S $\Omega(n^2)$ times [EE94]. However, in this construction, the constant factor is exponentially small in r.

A bound on $f(n,r)$ can be obtained by bounding the self-similarity of a point set as follows: partition the $\binom{r}{3}$ triplets of points in R into equivalence classes of similar triangles. The goal is to show that a constant fraction of the triplets in R belong to small classes.

Definition A point set R with r points is f-similar if at least $\frac{1}{2}\binom{r}{3}$ of the triplets in R belong to classes of size at most $f(r)$.

Proposition 6 *If R is cn^δ-similar, then for any point set S with n points, R matches into S at most $2cn^{3+\delta}/(r-2)^3$ times.*

Proof A triplet in R is said to be *good* if it belongs to a class of size at most cn^δ. Consider the set of all transformations which map R into S. Each triplet in S can be matched to a good triplet in R at most cn^δ times over all of these transformations. Thus, a good triplet from R matches to a triplet in S at most $\binom{n}{3}cn^\delta$ times. Since every time R matches into S, all $\frac{1}{2}\binom{r}{3}$ of the good triplets are matched, there can be at most $2\binom{n}{3}cn^\delta/\binom{r}{3} \le 2cn^{3+\delta}/(r-2)^3$ such matches. ■

Proposition 7 *Suppose that for every $r \le m$, there is a subset $R \subseteq M$ where $|R| = r$ and R is $O(r^\delta)$-similar. Then it can be determined if there is an occurrence of M in any scene S in time*

$$O\left(n^{2+\frac{1+\delta}{4}} m^{\frac{1}{4}}\right).$$

Proof Fix a particular r in the range $1 \le r \le m$ and let R be the subset of M of size r which is $O(r^\delta)$-similar. Using Proposition 7, when R is used to screen potential transformations, the running time of the alignment method is $O(n^2 r + (n^{3+\delta}/r^3)m)$. Choosing $r = (n^{1+\delta}m)^{\frac{1}{4}}$, minimizes this value, yielding a running time of $O(n^{2+(1+\delta)/4}m^{1/4})$. ■

The above bound yields an improvement over the bound of $O(n^2 m)$ from Section 2 when $n = o(m^{\frac{3}{1+\delta}})$. We know that no class ever can have more than m^2 triangles. Unfortunately, using $\delta = 2$ does not yield any improvement. Although we know that point sets can have as many as $\Omega(m^2)$ triangles in a single class, we do not know of any point sets in which more than a constant fraction of the triplets belong to large classes. This leaves open the possibility of showing that for all points sets, a constant fraction of the triplets belong to small classes. It can be proven that a set of r evenly spaced points on the line is $O(r)$-similar. We conjecture that this is true for every point set.

Conjecture 8 *Every set of r points in the plane is $O(r)$-similar.*

An interesting special case is when either the scene set S or the model set M are the points of a unit grid. We might expect such a point set to maximize the number of matches because of its regularity. When the model is a grid, we can prove the following bound for the self-similarity measure

Lemma 9 *Let R be an $r_1 \times r_2$ grid and let $r = r_1 r_2$. Then R is $O(r \log^4 r)$-similar.*

In order to prove Lemma 9, we will need the following lemma.

Lemma 10 *Let $\mathcal{T}$ be the set of all transformations which are compositions of translations, scaling, and rotation. Then if R is a grid of $r_1 \times r_2$ points and we define $r = r_1 r_2$,*

$$\sum_{t \in \mathcal{T}} \binom{|t(R) \cap R|}{3} \le 6r^4 \log^4 r.$$

Proof of Lemma 9 We will prove that there is a constant c such that there are no more than $\frac{1}{2}\binom{r}{3}$ triplets in classes with more than $cr\log^4 r$ triplets each.

Consider all the transformations such that R matches onto itself in more than three points. That is, let $\mathcal{T}$ be such that $t \in \mathcal{T}$ if and only if, $|t(R) \cap R| \geq 3$. Thus when t is applied to R, there are

$$\binom{|t(R) \cap R|}{3}$$

triangles in R which map on to a similar triangles in $t(R)$. Now let $\mathcal{C}$ denote the set of all classes of triplets in R. The following sum

$$S = \sum_{t \in \mathcal{T}} \binom{|t(R) \cap R|}{3}$$

is at least

$$\sum_{C \in \mathcal{C}} \binom{|C|}{2}.$$

Now suppose that there are more than $\frac{1}{2}\binom{r}{3}$ triplets in classes with more than $cr\log^4 r$ triplets each. Then

$$S \geq \frac{1}{2}\binom{r}{3} \frac{1}{cr\log^4 r} \binom{cr\log^4 r}{2} \geq c' r^4 \log^4 r.$$

c can be chosen so that $c' > 6$ which would contradict Lemma 10. ∎

Proof of Lemma 10 Let R be a unit grid, of $r_1 \times r_2$ points centered at the origin. First, we will fix the slope and scale factor of the transformation and then determine how many points match under all translations with this slope and scale. Let t be a transformation which consists of a rotation about the origin and a scale factor. For the time being, we restrict our attention to transformations where the scale factor is at least 1. A transformation for which the scale factor is less than one is simply the inverse of one where the scale factor is greater than one. Since $|t(R) \cap R| = |R \cap t^{-1}(R)|$, the sum which we are bounding will only double when all scale factors are considered.

The set $t(R) \cap R$ is either just the origin or forms a grid pattern. This grid pattern can be described by a pair (a, b) such that $1 \leq a \leq r_1$ and $1 \leq b \leq r_2$. The points in the intersection are of the form $i(a, b) + j(-b, a)$ where i and j are integers. This transformation can be obtained by taking a point in R located at (c, d), and rotating and scaling R about the origin so that the point is placed in location (a, b). Since the scale factor is at least 1, $c^2 + d^2 \leq a^2 + b^2$.

Let $G_{a,b}$ be the set of all points of the form $i(a, b) + j(-b, a)$. $G_{a,b}$ is an infinite grid with slope b/a whose points are spaced $\sqrt{a^2 + b^2}$ apart. $G_{a,b}$ intersects with R in at most $r/(a^2 + b^2)$ points. Since $R \cap t(R) \subseteq R \cap G_{a,b}$, we know that $|R \cap t(R)| \leq r/(a^2 + b^2)$.

How many translations of $t(R)$ are there such that the two sets R and $t(R)$ intersect in more than one point? As the translation of $t(R)$ is varied, the set of intersection points is either empty or maintains the same grid structure with the same scale and rotation, although the grid now may be translated. Thus, the intersection set is of the form $G_{a,b} + (u, v)$. There are at most $(a^2 + b^2)$ values for (u, v) which produce distinct grids which have a non-trivial intersection with R.

Now fix (u, v), say at $(0, 0)$. How many translations of $t(R)$ are there such that $t(R) \cap R$ has more than one point and is contained in $G_{a,b}$? This is the same as asking how many translations are there of $t(R)$ such that $t(R)$ and $(R \cap G_{a,b})$ intersect in more than one point. We can find a constant set Q of points in $t(R)$ and $R \cap G_{a,b}$ with the following property: for any translation of $t(R)$ such that its intersection with $R \cap G_{a,b}$ contains more than one point, at least one point in Q is contained in the intersection. Q consists of the the point in $t(R)$ closest to the center, the point in $R \cap G_{a,b}$ closest to the center, the points which are vertices of the convex hull of $t(R)$, and the points which are vertices of the convex hull of $R \cap G_{a,b}$. Since

$t(R)$ is a rectangular grid, there are only four vertices on its convex hull. Since $R \cap G_{a,b}$ is the intersection of a rotated grid with a rectangular area, there are at most eight vertices on its convex hull. Thus, there are at most 14 points in Q. The number of translations of $t(R)$ such that a point in Q is contained in the intersection is at most $14r$.

Thus, the sum we wish to bound is at most

$$\begin{aligned} & \sum_{a,b,c,d} \binom{r/(a^2+b^2)}{3} (a^2+b^2)(14r) \\ \leq & \sum_{a,b,c,d} \frac{r^4}{6(a^2+b^2)^3} 14(a^2+b^2) \\ \leq & \sum_{a,b,c,d} \frac{3r^4}{(c^2+d^2)(a^2+b^2)} \\ \leq & \sum_{a,b,c,d} \frac{3r^4}{abcd} \\ \leq & \ 3r^4 \log^4 r. \end{aligned}$$

The second inequality comes from the fact that we assumed that $c^2+d^2 \leq a^2+b^2$. We have an extra factor of two to include the transformations in which the scale factor is less than 1. Thus, we have bounded the sum by $6r^4 \log^4 r$. ■

Lemma 9 indicates that if the model is a grid, we can pick R to be a subgrid of the appropriate size and use it to screen potential matches for M. Choosing the optimal value for r gives the following corollary:

Corollary 11 *If the model is a grid, then it can be determined if there is an occurrence of the model in any scene in time $O(n^2(nm\log^4 m)^{1/3})$.*

This yields an improvement over the $O(n^2 m \log n)$ bound for alignment when $n = o((m/\log^2 m)^2)$. We conjecture that the grid is the worst case model and that the improvement can be obtained for any model.

When the scene is a unit grid, we do not need the self-similarity measure to improve the bound, but rather can prove the following lemma directly.

Lemma 12 *Any point set R with r points can match perfectly into an $n_1 \times n_2$ unit grid $O(n^2/r)$ times, where $n = n_1 n_2$.*

Choosing $r = \sqrt{m}$, we can find a match of any model with m points in the unit grid in time $O(n^2\sqrt{m})$. The most outstanding open problem remaining is that of better bounding the running times of RandomAlign and GeometricHash using refined bounds on self-similarity.

Proof of Lemma 12 Consider the pair of points in R that are the closest together. Call one o and the other p. Let q be the point in R which is farthest from o. It must be the case that $d(o,q)/d(o,p) = \Omega(\sqrt{r})$. Let ϕ be the angle between the line through o and p and the line through o and q. Now consider all matches of R to A where o is matched to some point $a \in A$.

Consider the area which is of dimension $(n_1/\sqrt{r}) \times (n_2/\sqrt{r})$ centered at a such that the rectangle is slanted at an angle of ϕ with respect to A. A is scaled so that two neighboring grid points are unit distance apart. If p is matched to a point outside of the rectangle, then the scale factor will be such that q will land outside of the grid A.

Thus, for each point $a \in A$, there are at most $(n_1/\sqrt{r})(n_2/\sqrt{r})$ points which p can match to if o matches to a. Thus, there are at most n^2/r matches of R to A. ■

5 Conclusion

This paper has explored the use of randomization with two basic matching techniques used in model-based recognition: alignment and geometric hashing. The analytical as well as empirical results of this study have shown that randomization is a very effective technique in reducing the running time of both of these methods, while very little is sacrificed in the accuracy of the methods.

Motivated by the observation that the empirical running time of the methods we have studied are much faster than the best theoretical bounds known for these algorithms, we have explored ways of improving these bounds. We have developed a combinatorial measure of the self-similarity of point sets and show how this notion could be used to improve the worst case bound for the alignment method. We conjecture that all point sets have self-similarity properties which would yield significantly improved bounds for point matching and we prove this property for grids of points.

Acknowledgements

We thank David Eppstein for useful discussions.

References

[AS92] N. Alon and J. Spencer. *The probabilistic method.* John Wiley and Sons, Inc., 1992.

[Che52] H. Chernoff. A measure of asymptotic efficiency for tests based in the sum of observations. *Annals of mathematical statistics*, 32:492–509, 1952.

[CGHKKK] L.P. Chew, M. Goodrich, D. Huttenlocher, K. Kedem, J. Kleinberg, D. Kravets. Geometric pattern matching under Euclidean motion. *Proceedings of the Fifth Canadian Conference on Computational Geometry,* 1993. p. 151-6.

[EE94] G. Elekes and P. Erdős. Similar Configurations in Planar Point Sets. Manuscript.

[FB81] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[Fel50] W. Feller. *An introduction to probability theory and its applications.* John Wiley and Sons, 1950.

[GH90] W.E.L. Grimson and D.P. Huttenlocher. On the sensitivity of geometric hashing. In *Proceedings of the 3rd International Conference on Computer Vision*, pages 334-338, 1990.

[HH92] J.E. Hopcroft and D.P. Huttenlocher. *Geometric invariance on computer vision*, chapter 18, pages 354–374. MIT Press, 1992.

[HU87] D.P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proceedings of the 1st International Conference on Computer Vision*, pages 102–111, 1987.

[HU90] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment. *International Journal of Computer Vision*, 5(2):255–274, 1990.

[LSW88] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. Object recognition by affine invariant mapping. In *Proceedings of Conference on Vision and Pattern Recognition*, pages 335–344, 1988.

[LW88] Y. Lamdan and H.J. Wolfson. Geometric hashing: a general and efficient model-based recognition scheme. In *Proceedings of the 2nd International Conference on Computer Vision*, pages 238–249, 1988.

[LW91] Y. Lamdan and H.J. Wolfson. On the error analysis of geometric hashing. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 22-27, 1991.

[Wol90] H.J. Wolfson. Model based object recognition by 'geometric hashing'. In *Proceedings of the 1st European Conference on Computer Vision*, pages 526–536, 1990.