Electronic Notes in Theoretical Computer Science 46 (2001)
URL: `http://www.elsevier.nl/locate/entcs/volume46.html` 15 pages

A Practical, Globally Optimal Algorithm for Geometric Matching under Uncertainty

Thomas M. Breuel [1]

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Abstract

Geometric matching under uncertainty is a long-standing problem in computer vision. This paper presents a simple and efficient branch-and-bound algorithm for finding globally optimal solutions to geometric matching problems under a wide variety of allowable transformations (translations, isometries, equiform transformations, others) and a wide variety of allowable feature types (point features, oriented point features, line features, line segment features, etc.). The algorithm only requires an implementation of the forward transformation (model-to-image) and an error model to be supplied. Benchmarks and comparisons of the algorithm in comparison with alignment and Hough transform methods are presented.

1 Introduction

Many problems in computer and machine vision involve matching geometric models to image data under geometric uncertainty. Such problems can be described as follows. Let a model consist of a collection of geometric primitives (points, line segments, etc.), generally referred to as "features". Images are assumed to be related to model by a geometric transformation T of the model features (e.g., a translation and rotation of the model), the deletion (occlusion) of some features, the addition of noise of a known distribution to the feature locations in the image, and the addition of random background features (clutter) not derived from the model. Let us assume for now that the noise is bounded by some error bound ϵ. Under simple additional assumptions, a maximum likelihood or maximum a-posterior interpretation of the image can be found by maximizing the number of image features that can be brought into correspondence to model features under the given error bound

[1] Email: `tbreuel@parc.xerox.com`

and under some model-to-image transformation T. Object recognition problems are therefore commonly formalized as the geometric matching problem of identifying the transformation T in a given space of possible transformations that brings a maximum number of model features into correspondence with image features under a given error bound ϵ.

A wide variety of algorithms have been developed for solving these kinds of geometric matching problems. This paper cannot hope to give a complete survey of these techniques, but the following describes some major ideas in the field that are relevant to the algorithm described in this paper.

Recognition by alignment[15] works by repeatedly selecting a small collection of model features and putting them in correspondence with a collection of image features. The size of these collections is determined by the minimal number of feature correspondences needed in order to determine a transformation uniquely. Since the transformation computed from the correspondence between model and image features is computed based on data that has been corrupted by location error, the transformation determined in this way will not necessarily be the transformation that maximizes the overall number of feature correspondences, however. Therefore, recognition by alignment is only an approximation or a heuristic for optimal geometric matching.

Correspondence search[12] is a method closely related to recognition by alignment. However, rather than using a minimal number of correspondences, possible correspondences between image and model features are explored in a search tree, and for a given set of correspondences, an overall "good" transformation is determined, for example using a least square method. If run to completion, such a search algorithm will find the optimal match between an image and a model. However, such search methods are subject to combinatorial explosion.

Several provably polynomial time geometric matching algorithms have been described in the literature (e.g., Cass[9]). There are a number of ways of looking at, and implementing, those algorithms, but in terms of complexity, they appear to be equivalent to sweeping or exploring a geometric arrangement[11] created by the constraint sets[1] implied by correspondences between model features and image features. Directly applied, such methods do not appear to be practical. However, the insights they are based on form the basis for the algorithm presented in this paper.

Pose clustering techniques[20] are based on examining the transformations ("poses") implied by many different hypothesized correspondences between image and model features. Transformations that bring many image and model points into correspondence under given error bounds will tend to cluster in the space of transformations. Because error bounds in the image do not translate directly into easily definable error bounds in transformation space, however, and thus such approaches are only heuristic.

Hough transforms (reviewed in [16]) are another approach to geometric matching closely related to pose clustering (and predating it by many years).

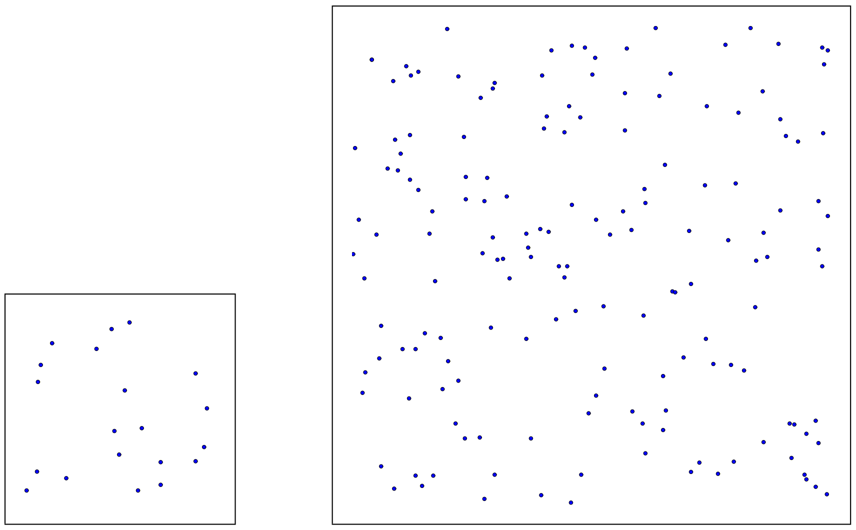
Fig. 1. A simple instance of the geometric matching problem. The image on the right contains a subset of 10 points from the image on the left, translated, rotated, and each displaced by a random displacement of less than 5 pixels.

Hough transforms can be viewed as performing pose clustering using various simple binning methods in the space of transformations. Hough transforms are easy to implement and quite fast; with careful tuning, they can give reasonably reliable answers. However, like other pose clustering techniques, Hough transform methods do not model location error with complete accuracy. Furthermore, unlike most other geometric matching techniques, Hough transforms also do not enforce the constraint that a single model feature gives rise only to a single image feature. As a result, Hough transforms can be quite susceptible to both false positives and false negatives.

Roughly speaking, these methods fall into two categories: approaches that guarantee correct solutions but have high complexity and may be difficult to implement, and approaches that are fast but heuristic, in the sense that they cannot guarantee finding optimal solutions.

The RAST (Recognition by Adaptive Subdivision of Transformation Space) family of algorithms [3] combines usable performance with a guarantee of finding geometrically well-defined solutions. RAST algorithms have been described for line finding under bounded error[6], for geometric matching under equiform transformations [2], and for geometric matching of point features under translation and rotation [8]. Other authors have used RAST-like algorithms for matching under Gaussian error [17]. Branch and bound style algorithms have received more attention in computer vision recently (e.g., [13,18,19]); we will return to a comparison of these approaches in the conclusions.

2 Inputs

There are two kinds of inputs to the algorithm. First, there is the data-independent portion: a function that computes the parameterized geometric transformation from model to image features, and a function that evaluates the quality of match between a single transformed model feature and an image feature. Second, there is the data-dependent portion: the actual coordinates of the model and image features.

For simplicity, and without loss of generality, let us assume that the set $\mathcal{T}$ of possible transformations T is parameterized by elements of the unit hypercube $[0,1]^D$. We will use T to refer both to the transformation itself and its parameterization. For example, for matching under isometries (translation and rotation) the parameter space for T would be $[0,1]^3$. For a collection of image and model points whose distance from the origin is each bounded by 512 pixels, we might choose a parameterization of the transformation T as follows:

$$\begin{aligned} (1)\quad \begin{pmatrix} x' \\ y' \end{pmatrix} &= T \begin{pmatrix} x \\ y \end{pmatrix} \\ (2)\quad &= \begin{pmatrix} \cos 2\pi T_3 & -\sin 2\pi T_3 \\ \sin 2\pi T_3 & \cos 2\pi T_3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1024\, T_1 \\ 1024\, T_2 \end{pmatrix} \end{aligned}$$

We make no special assumptions about this parameterization other than that its derivative should be bounded for any transformation and any fixed bound on the coordinates of the model features: for $T \in [0,1]^D$ and $x, y < \text{const}$,

$$(3)\qquad \frac{\partial x'}{\partial T_i}(T,x,y) < \text{const} \quad \text{and} \quad \frac{\partial y'}{\partial T_i}(T,x,y) < \text{const}$$

The other data-independent ingredient to the algorithm is a function that computes matches under our error model. For concreteness in this discussion, let us assume a bounded error model, although other error models (like Gaussian) can be incorporated easily and with little change. The feature match function b takes as input a transformed model feature TM, an image feature I, and an error bound ρ, and computes a match score. We require the feature match function to be monotonic:

$$(4)\qquad b(TM, I, \rho) \leq b(TM, I, \rho') \text{ if } \rho < \rho'$$

This monotonicity condition is satisfied (and easily verified) for all commonly used match criteria, including matching under bounded error, matching under any metric, and matching under Gaussian error. In the case of point features, $b(TM, I, \rho)$ might simply be defined as the indicator function for the predicate $||TM - I|| < \rho$, i.e., a function that assumes the value 1 if the distance between the transformed model feature TM and the image feature I is less than ρ, and 0 otherwise. In the case of line segment features, another common feature used in computer vision, $b(TM, I, \rho)$ might be defined as the total length of the

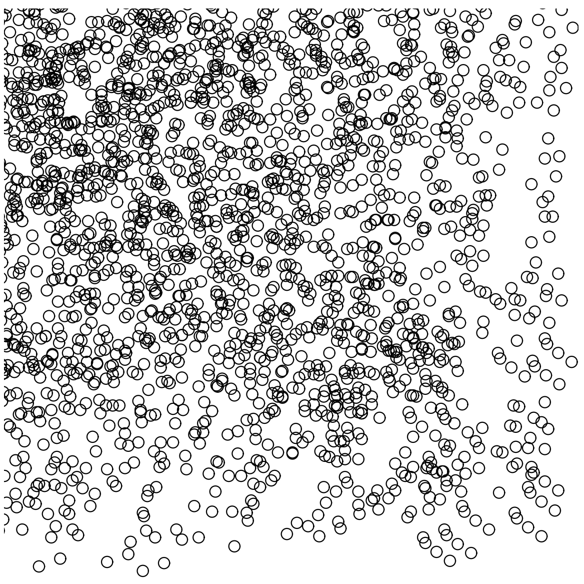

Fig. 2. An example of the arrangement generated by the constraint sets in a matching problem under translation only.

subsegment of the image line segment I that falls within a distance of ρ of the transformed model line segment TM. Note that such a measure still satisfies the monotonicity condition. Together with the feature match function, we also assume (for bounded error matching) a choice of error bound ϵ. Observe that $b(TM, I, \rho)$ will be evaluated for values of ρ different from the chosen error bound ϵ.

The data-dependent input to the algorithm is a set of model features $\mathcal{M} = \{M_1, \ldots, M_m\} \subseteq R^{D_M}$ and a set of image features $\mathcal{I} = \{I_1, \ldots, I_n\} \subseteq R^{D_I}$. In the case of matching points under isometric transformations of the plane, both image and model points are points in R^2.

Given the feature match function b and the sets of image and model features, the overall quality of match Q of a transformation T is given by:

$$Q(T) = \sum_{i=1\ldots m} \sum_{j=1\ldots n} b(T\,M_i, I_j, \epsilon) \tag{5}$$

The task of a geometric match algorithm as defined in this paper is to optimize this quality of match over all possible transformations:

$$T_{\max} = \arg \max_{T \in [0,1]^D} Q(T) \tag{6}$$

3 The Algorithm

We will first describe the matching algorithm and discuss the geometry and complexity briefly in later sections. The algorithm is a best first search through a recursive subdivision of the parameter space of transformations. For simplicity of exposition, let us assume a three dimensional parameter space. As the recursive subdivision, for concreteness, let us choose a data-

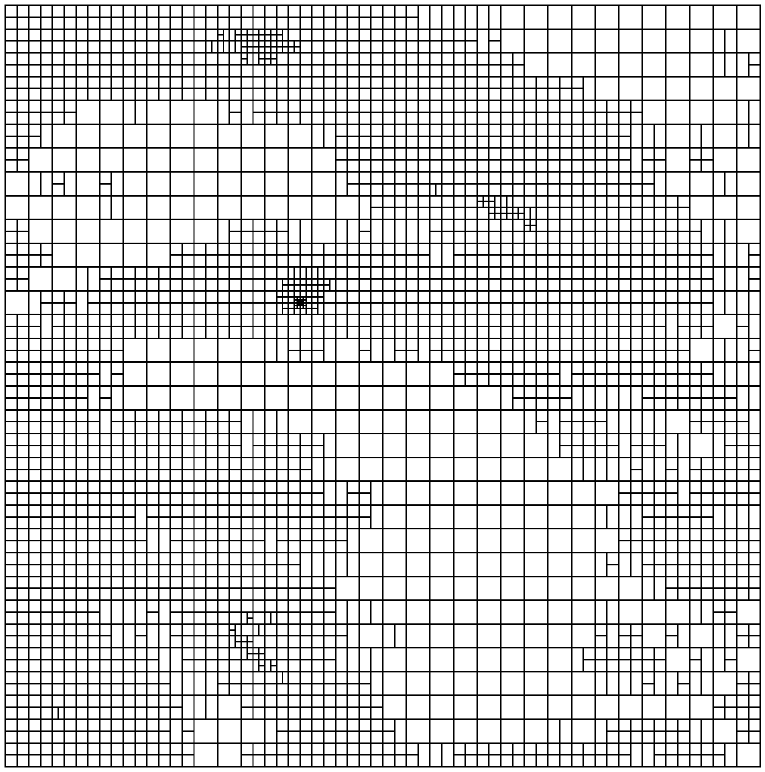

Fig. 3. The subdivision of transformation space explored during an actual run of the algorithm. Transformation space in this example is three dimensional, two translational components and one rotational component, but the space has been projected down to two dimensions along the rotational dimension. The location of the solution is recognizable as the densely explored area toward the upper left portion of the space.

independent kD-tree like binary subdivision of transformation space. At the base of the tree is the complete parameter space $R_1 = [0,1]^D$. For the recursive step, if we are looking at a node in the search space representing the region $R_r = [l_1, h_1] \times \ldots \times [l_D, h_D]$. Let $m = \arg\max_i \; h_i - l_i$ be the largest dimension of R_r. We split the region along that dimension. The two child nodes then are:

$$R_{2r} = [l_1, h_1] \times \ldots \times [l_m, \frac{(h_m + l_m)}{2}] \times \ldots \times [l_D, h_D] \tag{7}$$

$$R_{2r+1} = [l_1, h_1] \times \ldots \times [\frac{(h_m + l_m)}{2}, h_m] \times \ldots \times [l_D, h_D] \tag{8}$$

The subdivision of transformation space (projected down to two dimensions) from an actual run is shown in Figure 3.

For each of these regions R_r in transformation space that we expand, we compute an upper bound on the quality of match that any transformation $T \in R_r$ can generate. We compute this bound as follows.

For each model feature M_i, we can compute an upper bound δ on the distance $\delta_{\min} = \max_{T,T' \in R_r} ||TM_i - T'M_i||$. Note that δ can be a function of

R_r as well as the model feature M_i in question, $\delta = \delta(R_r, M_i)$ Also note that because of the bounded derivative property that we required above, $\delta_{\min} \to 0$ as $\operatorname{diam}(R_i) \to 0$, and we require that the upper bound $\delta(R_i, M_i) \to 0$ we choose also approaches 0 as $\operatorname{diam}(R_i) \to 0$. We can derive $\delta(R_r, M_i)$ manually from the analytic form of the transformation $p' = T\,p$, or we can compute it automatically by symbolic differentiation, numerical differentiation, or random sampling. (Such automatic derivations can be simplified and speeded up somewhat by further bounding $\delta(R_r, M_i)$ from above by $\delta(R_r, M)$ for $||M|| < \text{const.}$)

By simple geometry (the triangle inequality), we are guaranteed that $b(TM + v, I, \delta + \epsilon) \geq b(TM, I, \epsilon)$ if $||v|| < \delta$. Using this and monotonicity of the feature match function, we obtain $b(T'M, I, \delta + \epsilon) \geq b(TM, I, \epsilon)$ for all $T, T' \in R_i$, where δ is computed as above. Furthermore,

$$\max_{T \in R_i} Q(T) = \max_{T \in R_i} \sum_{i=1\ldots n} \sum_{j=1\ldots m} b(TM_j, I, \epsilon) \tag{9}$$

is bounded from above by

$$Q(T_0) = \sum_{i=1\ldots n} \sum_{j=1\ldots m} b(T_0 M_j, I, \delta + \epsilon) \tag{10}$$

for any $T_0 \in R_i$ because the terms of the sum are individually bounded from above. We call this upper bound $\hat{Q}(R_i)$. With these preliminaries, we have now the ingredients for the geometric matching algorithm:

Algorithm 1

```
1: Initialize the priority queue to the region of all transformations $R_1$ and an upper bound of $+\infty$.
2: While the priority queue is non-empty, extract the element with the highest priority (if there are multiple elements with equal priority, prefer the one with a larger depth $d = \lfloor \log r \rfloor$); call this element $R_r$.
3: If $R_r$ determines a solution to desired/machine accuracy, accept it as a solution and finish the search.
4: Split $R_r$ into its two child regions $R_{2r}$ and $R_{2r+1}$.
5: For each child region, compute $\hat{Q}(R_{2r})$ and $\hat{Q}(R_{2r+1})$.
6: Enqueue $R_{2r}$ with priority $\hat{Q}(R_{2r})$ and $R_{2r+1}$ with priority $\hat{Q}(R_{2r+1})$.
7: Continue at Step 2.
```

Described at this level of generality, the algorithm is quite similar to the algorithm described in [8]. A naive implementation might simply evaluate the two sums in Equation 10 directly. However, this would be very inefficient. This inefficiency could be partially remedied by using a point location data structure, as proposed in [8] for geometric matching problems and [19] for geometric primitive detection using RAST algorithms. However, a better approach is to keep track of model and image feature correspondences during the search itself. Because of the required monotonicity property, if $\hat{Q}(R_r)$ is zero, it will be zero for all children of R_r as well. Therefore, we only need to keep track of image and model features that actually result in non-zero

number of trials	2190
avg. # matches missed by alignment	1.0
fraction of trials with suboptimal alignment results	68.0%
fraction of trials with incorrect Hough results	83.7%

Fig. 4. Summary of the errors made by alignment and Hough transform methods relative to the geometrically optimal solution. For alignment, a solution was counted as "missed" if the transformation mapped fewer model features within the given error bounds of an image feature than the geometrically optimal solution. For the Hough transform, a much less stringent performance measure was used: a result was counted as "correct" if its translational component was within 2ϵ of the actual translation.

contributions to $\hat{Q}$. In practice, we do this by associating with each region R_r a list of pairs of model and image features that make non-zero contributions to $\hat{Q}(R_r)$; we refer to these lists as *matchlists*. As R_r shrinks, these matchlists themselves shrink. During each computation of $\hat{Q}(R)$ for successively smaller R, only image-to-model correspondences need to be considered that actually fell within the error bounds of the parent of R. This approach can be viewed as incorporating the construction and use of a point-location data structure directly into the search for an optimal solution.

4 Geometry and Complexity

To understand the performance and complexity of this algorithm, we need to look at the geometry of transformation space This paper does not attempt to provide a complete complexity analysis, but rather merely a description of the underlying geometry and some intuition of what the implications are for complexity. For a more detailed exposition than possible here, the reader is referred to the literature on both geometric matching and the computational geometry of arrangements (e.g., [10]). There are m model features and n image features. Hence, there are mn possible correspondences between model features and image features. Pick a single model feature M_i and a single image feature I_j. Now consider the feature match function as a function of the transformation T:

$$b_{ij}(T) = b(TM_i, I_j, \epsilon) \tag{11}$$

For simplicity of exposition, let us assume matching of point features under a bounded error model. In that case, $b_{ij}(T)$ only assumes the values 1 (if T maps model feature M_i inside the error bound ϵ from image feature I_j) or 0 otherwise. We can then consider the function $b_{ij}(T)$ to be the indicator function of a subset of transformation space; let us call this subset T_{ij}. In the

#image features	RAST time (in seconds)	alignment time (in seconds)	ratio RAST/alignment	Hough time (in seconds)
20	2.0	0.4	5	0.8
40	5.5	1.0	5.5	1.5
60	10.1	1.8	5.6	2.4
80	15.7	2.9	5.4	3.6
110	26.9	5.3	5.1	6.2
160	48.8	11.2	4.4	12.7

Fig. 5. Comparative running times of the RAST, alignment, and Hough transform methods on images with different numbers of features.

case of bounded error recognition, T_{ij} is referred to as a *constraint set* (e.g., [1]).

The collection of all T_{ij} form an *arrangement* in transformation space $\mathcal{T}$. By an "arrangement", we mean the collections of all possible subsets of transformation space that can be derived by intersections of any number of T_{ij}. We refer to these subsets as *cells* of the arrangement. An example of such an arrangement is shown in Figure 2 for the case where the space of all possible transformations consists of only translations. An analogous picture for the case of isometric transformations would consist of small, interpenetrating cylinders twisting through a three-dimensional cube; In the case of isometric transformations, if rotations are parameterized along the z-axis, each slice through the cube through a plane parallel to the xy-plane would look similar to Figure 2. It is easy to see that $Q(T)$ is constant over each cell. It is this arrangement that is explored by the data-independent space partitioning tree defined in Equation 7ff.

A formal average case analysis is beyond the scope of this paper, since it would involve a statistical analysis of geometric arrangements, a difficult subject. To demonstrate practicality of the algorithm, we rely on actual performance measurements in experiments (below). An informal average case analysis of a closely related problem can be found in [2] and suggests that the computational complexity of RAST-type algorithms is similar to the computational complexity of alignment methods.

5 Experiments

The algorithm described above was implemented for the case of matching unlabeled, unoriented point features under different kinds of transformations (translation, isometric, equiform). This is actually the most difficult feature

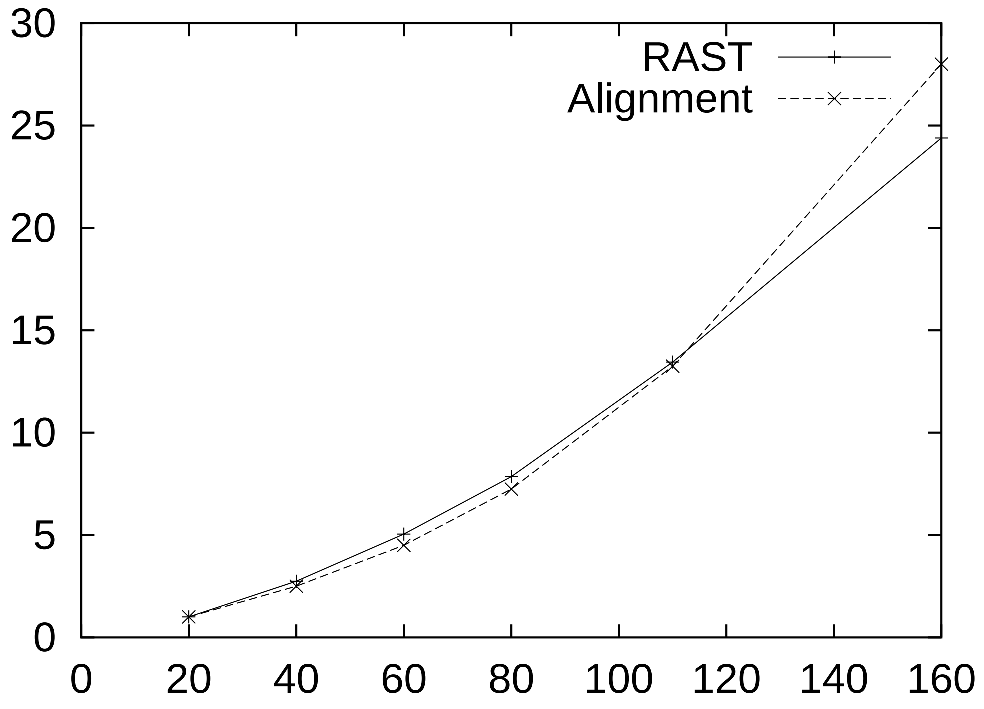

Fig. 6. A plot of the relative performance of RAST and alignment methods for geometric matching. The plots have been normalized to 1 for the running time. This shows that the RAST algorithm scales approximately like alignment over the range of parameters considered.

type to use since each feature by itself carries little information; using line segment or edge features reduces the running time and complexity of RAST algorithms, as well as alignment and Hough transform methods, relative to the unlabeled, unoriented point feature case; results from such experiments are not shown here. Benchmarks on randomly generated data were carried out to compare the performance of this algorithm with matching by alignment and matching using the Hough transform (performance on features derived from real image data is similar). The goals of these experiments were to determine how this algorithm scales compared to alignment and Hough transform methods and how large the "constant factors" are in the relative running times of the different methods. In addition, since both recognition by alignment and the Hough transform do not guarantee finding geometrically optimal matches, these experiments measure how often alignment or Hough transform methods return suboptimal results.

In these experiments, models consisted of 20 points randomly and uniformly drawn from the region $[-100, 100] \times [-100, 100]$. Given a randomly generated model, o generate the image, the system randomly picked a rotation angle in $[0, 2\pi)$ and a translation from $[100, 400] \times [100, 400]$ and transformed the model with this transformation. Subsequently, 10 of the 20 transformed model features were deleted (simulating occlusions and sensor failure) and the remaining 10 model features were randomly perturbed by uniformly distributed vectors of magnitude less than the error bound $\epsilon = 5$. To simulate clutter and context, a variable number of points drawn randomly and uniformly from the region $[0, 512] \times [0, 512]$ were added to the image. In a final step, the order of all image and model features was randomize. In this way, a collection of 3000 test cases was generated, with models consisting of 20 feature points and images consisting of between 20 and 160 feature points.

This was the input given to the three different algorithms that were tested: RAST, alignment, and the Hough transform. The RAST algorithm was implemented as described in the previous section in about 300 lines of C++ code (plus supporting code for the testing framework). The implementation itself reflects the independence of the search algorithm from the geometry: the geometry of the matching is specified as a "traits" class, defining only methods for splitting a region in transformation space into subregions, for transforming a model feature given a subregion in transformation space (giving a location and an error bound), and a method for measuring the quality of match between an image feature and a transformed model feature. The recognition algorithm itself operates entirely in terms of this abstract interface. For memory management, it used a conservative garbage collector. Since memory management in the RAST algorithm could be implemented fairly easily in a stack-like manner using memory pools, resulting in virtually no cost to memory allocation/deallocation, this implementation imposes a significant amount of unnecessary memory management overhead, probably in the range of 30%-50%, relative to the Hough and alignment implementations described below. In different words, the relative performance of the RAST algorithm to the alignment and Hough methods could probably be improved by up to a factor of 2 if the code is modified to use a more efficient memory management strategy.

For the alignment algorithm, the system iterated through all pairs of model features. For each pair of model features M_i, M_j, all pairs of image features $I_{i'}$, $I_{j'}$, where $|\ ||I_{i'} - I_{j'}|| - ||M_i - M_j||\ | < 2\epsilon$ (other features cannot be aligned under isometric transformations). This process was speeded up using an efficient 1D point location data structure (a 1D trie), so that only m^2nl rather than m^2n^2 alignments need to be considered, where l is the number of image features $I_{j'}$ that are found at the correct distance from $I_{i'}$. Once the transformation was calculated, the remaining model features were subjected to the same transformation and tested for whether they fell within the given error bound of some image feature. This process was speeded up using an efficient 2D point location data structure (a 2D trie) for the image features.

For the Hough transform, pairs of model features and pairs of image features were put into correspondence. This resulted in a single transformation (similar to the transformation found in the alignment algorithm) and the resulting transformation parameters were computed (translation and rotation). These three transformation parameters (two translational and one rotational parameter) were scored in a three-dimensional accumulator array. The bin size for the translational components was chosen to be the same as the error bound ϵ. The bin size for the rotational component was chosen to be 0.05. The optimal match was considered to be represented by the transformation associated with the center of the bin containing the largest number of votes.

Figure 4 shows the rates at which matching by alignment and by Hough transform returned suboptimal solutions. In fact, recognition by alignment found only a suboptimal solution (i.e., missed one or more feature correspondences) in 68.0% of the trial runs. On average, recognition by alignment missed approximately one feature match per trial (i.e., the transformation determined by alignment would fail to put, on the average, one of the model features into correspondence with an image image, while an optimal transformation did).

The transformation determined by the simple Hough transform implementation is almost never accurate enough to result in a good match score; Hough transforms are often used as a preprocessing filter. Therefore, a more lenient way of evaluating the Hough transform was chosen: the transformation returned by the Hough transform was considered correct if its translational component fell within 2ϵ of the translational component of the actual transformation that was used to generate the image features from the model features (the rotational component was not required to match). Nevertheless, even under this measure, the Hough transform returned suboptimal or incorrect results in 83.7% of the cases. The high error rate of the Hough transform in these experiments is not difficult to understand: it results from quantization and aliasing of the Hough space and the lack of modeling of other effects. More sophisticated and accurate versions of the Hough transform are known that model such effects more accurately and probably would result in lower error rates in these kinds of experiments. However, such extended Hough methods are also more complex to implement and tend to have higher runtime overhead. The simple Hough implementation provided in this paper is given merely as a fairly easy to understand baseline.

The absolute and relative running times of the different geometric matching algorithms is shown in Figure 5. The Hough transform and alignment perform roughly equally, With the current implementation, the RAST algorithm takes approximately five times as long as the alignment or Hough methods (as noted above, with better memory management in the RAST code, this could probably be reduced by up to a factor of 2 with no significant changes to the code). The relative performance of the RAST and the alignment methods is also illustrated in Figure 6, showing close agreement in the scaling of runtime with problem size between the two methods over the range of image sizes tested.

6 Discussions and Related Work

This paper has presented a simple, practical algorithm for finding globally optimal solutions to geometric matching problems. The algorithm is based on a branch and bound exploration of the space of possible transformations. For the computation of upper bounds, it uses *matchlists*. That is, associated with each subregion of transformation space considered by the algorithm, it maintains a list of correspondences that are potentially consistent with transformations in that subregion. The idea of matchlist based branch-and-bound geometric matching was introduced in [3,2] for the case of matching under equiform transformations using Baird's linear representation [1], and later extended to arbitrary transformations [5] and parametric models [6]. For its application, the algorithm requires only that a user supply an implementation of the forward imaging transformation and a function evaluating the quality of match between a single transformed model feature and a single image feature. The user can pick convenient parameterizations (linear or non-linear) of transformations, making the implementation of matching under, for example, isometric transformations or translation+scale easy. This paper has presented a more complete exposition of the algorithm for the case of isometric (translation+rotation) transformations in the plane, together with experimental results analyzing its performance and comparing it with other commonly used approaches.

Matchlist-based approaches are distinguished from other branch and bound methods for geometric matching in that they do not require a point location data structure. Other proposed approaches to geometric matching use Voronoi diagrams [14] or point location data structures [13,18,19] in the computation of upper bounds. The use of point location data structures has a variety of disadvantages, including increased implementation complexity, and possibly the introduction of quantization errors (with some data structures). The experimental results presented in this paper show that a matchlist-based approach is practical; further experiments demonstrate [7], in fact, that a matchlist based approach is often more efficient than a point location based approach.

While some theoretical questions remain about the worst case complexity of the algorithm, as well as a formal determination of its average case complexity, the paper has demonstrated experimentally that when applied to difficult, randomly generated matching problems, the algorithm exhibits the same computational complexity as recognition by alignment and runs within a small constant factor of alignment algorithms.

Another novel result is the comparison between the quality of match found by alignment methods, Hough transform methods, and the optimal solution. These experimental results show that commonly used geometric matching methods like alignment and Hough transforms very frequently return suboptimal solutions. Such methods attempt to overcome these limitations by additional "verification", "backprojection" or other fixup steps (e.g., [12]).

However, such post-processing is usually merely heuristic and still does not guarantee optimality of the results.

While there is a large number of heuristic geometric matching algorithms, among the small number of geometric matching algorithms described in the literature that can guarantee returning geometrically optimal results, the algorithm presented in this paper appears to be the simplest and most efficient to date. Previous members of this algorithmic family have already found applications in handwriting recognition, 3D model-based recognition [4], and document analysis. The improvements presented in this paper simplify the algorithm further and make it more widely applicable. Furthermore, the new experimental results presented in this paper demonstrate that the algorithm scales comparably to existing methods but returns better match results in a large fraction of the cases. Given the moderate extra runtime cost and simplicity of the algorithm, the fact that it does not require separate hypothesis generation and verification steps, the fact that it requires no "parameter tuning" in order to apply to particular problems, and the fact that it is guaranteed not to lose optimal solutions, it should prove to be useful in many more applications in computer vision and document analysis.

References

[1] Baird, H. S., "Model-Based Image Matching Using Location," MIT Press, Cambridge, MA, 1985.

[2] Breuel, T. M., *Fast Recognition using Adaptive Subdivisions of Transformation Space*, in: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 1992, pp. 445–451.

[3] Breuel, T. M., "Geometric Aspects of Visual Object Recognition," Ph.D. thesis, Massachusetts Institute of Technology (1992), also available as MIT AI Lab TR# 1374.

[4] Breuel, T. M., *View-Based Recognition*, in: *MVA '92, IAPR Workshop on Machine Vision Applications*, 1992, pp. 29–32.

[5] Breuel, T. M., *Recognition by Adaptive Subdivision of Transformationn Space (RAST) – Practical Experiences and Comparisons with the Hough Transform*, in: *IEE Colloquium on Hough Transforms, Savoid Place, London*, 1993.

[6] Breuel, T. M., *Finding Lines under Bounded Error*, Pattern Recognition **29** (1996), pp. 167–178.

[7] Breuel, T. M., *Branch and Bound Algorithms for Geometric Matching*, In Preparation (2001).

[8] Breuel, T. M., *Implicit manipulation of constrain sets for geometric matching under 2d translation and rotation*, in: *Proc. 12th Scandinavian Conference on Image Analysis*, 2001.

[9] Cass, T., *Polynomial-time geometric matching for object recognition*, International Journal of Computer Vision **21** (1997), pp. 37–61.

[10] Edelsbrunner, H., "Algorithms in Combinatorial Geometry," Springer Verlag, 1987.

[11] Edelsbrunner, H. and L. J. Guibas, *Topologically sweeping an arrangement*, Technical Report UNICDCS-R-86-1255, Univ. Illinois at Urbana-Champaign, USA (1986).

[12] Grimson, E., "Object Recognition by Computer," MIT Press, Cambridge, MA, 1990.

[13] Hagedoorn, M. and R. C. Veltkamp, *Reliable and efficient pattern matching using an affine invariant metric.*, Technical Report RUU-CS-97-33, Dept. of Computing Science, Utrecht University (1997).

[14] Huttenlocher, D. P. and W. J. Rucklidge, *A Multi-Resolution Technique for Comparing Images Using the Hausdorff Distance*, in: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 1993, pp. 705–706.

[15] Huttenlocher, D. P. and S. Ullman, *Object Recognition Using Alignment*, in: *Proceedings of the International Conference on Computer Vision* (1987), pp. 102–111.

[16] Illingworth, J. and J. Kittler, *A Survey of the Hough Transform*, Computer Vision, Graphics and Image Processing **44** (1988), pp. 87–116.

[17] Jurie, F., *Solution of the simultaneous pose and correspondence problem using Gaussian error model*, Computer Vision and Image Understanding: CVIU **73** (1999), pp. 357–373.
URL `citeseer.nj.nec.com/284653.html`

[18] Mount, D., N. Netanyahu and J. Le Moigne, *Efficient algorithms for robust feature matching*, Pattern Recognition **32** (1999), pp. 17–38.

[19] Olson, C. F., *Locating geometric primitives by pruning the parameter space*, Pattern Recognition **34** (2001), pp. 1247–1256.

[20] Stockman, G., *Object recognition and localization via pose clustering*, Computer Vision, Graphics, and Image Processing **vol.40, no.3** (1987), pp. 361–87.