Performance Comparison of Ten Variations on the Interpretation-Tree Matching Algorithm

Robert B. Fisher

Dept. of Artificial Intelligence, University of Edinburgh
5 Forrest Hill, Edinburgh EH1 2QL, Scotland, United Kingdom

Abstract. The best known algorithm for symbolic model matching in computer vision is the *Interpretation Tree* search algorithm. This algorithm has a high computational complexity when applied to matching problems with large numbers of features. This paper examines ten variations of this algorithm in a search for improved performance, and concludes that the non-wildcard and hierarchical algorithms have reduced theoretical complexity and run faster than the standard algorithm.

1 Introduction

The most well-known algorithm for symbolic model matching in computer vision is the *Interpretation Tree* (IT) search algorithm[7]. The algorithm searches a tree of potential model-to-data correspondences, which is a key problem in model-based vision, and is usually a preliminary to pose estimation, identity verification or visual inspection. This algorithm has the potential for combinatorial explosion, even with techniques for limiting the search[7]. This paper compares ten extensions (mainly found in the published literature) to the standard IT algorithm that have the potential to reduce the search space. The results of the paper show that several of the variations produce improved performance in both theory and as applied to real data.

We follow the standard IT model[8]:

- There are M model features in the model.
- On average, $p_v M$ of these are visible in the scene. In 2D scenes, $p_v \doteq 1$ and, in 3D scenes, $p_v \doteq 0.5$ as about half of the features are back-facing.
- Of the visible model features, only p_r of these are recognizable forming $C = p_r p_v M$ correct matchable data features.
- There are also S spurious features and thus $D = C + S$ data features.
- The probability that a randomly chosen model feature matches with an incorrect random data feature is p_1.
- The probability that a random pair of model features is consistent with an incorrect random pair of data features (given that the individual model-to-data pairings are consistent) is p_2.
- An acceptable set of model-to-data pairings must have at least $T = \tau p_v M$ non-wildcard correspondences ($\tau \in [0, 1]$).

2 The Algorithmic Variations

Geometric Matching: Once enough model-to-data pairings have been formed, it is (usually) possible to estimate a pose[2]. Then, the exponential portion of the search algorithm stops for that branch. The pose estimate allows the prediction of the image position of unmatched model features including which are back-facing and hence not visible[4]. If a model feature is visible, then direct search is used to find data features whose position is consistent with the predicted model feature. The effort required to do each comparison is assumed to be comparable to that of standard algorithm's testing.

If spatial indexing of data features is possible, then the direct search phase need only match against data features directly indexed, instead of all features. We assume here that spatial indexing is sufficiently good that only 1 incorrect feature is selected. We investigated algorithms that required 2 (e.g. for 2D) and 3 (e.g. for 3D) matches before going to geometric matching.

Alignment Methods: After several levels of the interpretation tree are explored, a model pose can be estimated and used to predict the position of the remaining unmatched model features. Data features near the prodicted positions are then used for subsequent levels of the IT. Here, the IT is searched by expanding model levels, rather than data levels. In the experiments below, we assume that direct search starts after 2 non-wildcard features are matched and the number of candidate features found is as described above.

Subcomponent Hierarchies: Suppose that the $M = K^L$ model features can be decomposed into K^{L-1} primitive subcomponents each containing K features, each matched to the D data features in the standard way. Then, each of the subcomponents are grouped into K^{L-2} larger models, each containing K^2 features, and so on hierarchically until we have one top-level model containing all K^L model features. Let each group at each level now define a new type of model feature representing its particular set of subcomponents. The hierarchical matching algorithm[6] generates hypotheses of these submodel types, by combining matched sets of features (i.e. submodels) from the next lower level. The algorithm is a top-down matching process, in which the largest possible matches are always generated first, and previous successes are recorded to limit computation when back-tracking. Consistency is checked using the standard IT criteria.

Subcomponent Hierarchies Using Reference Frame Consistency: When a new hypothesis is tested for consistency, if the two subcomponent hypotheses have sufficient features matched that their poses have been estimated, then only the poses of the subcomponent hypotheses are checked for consistency relative to the pose of their "parent" hypothesis.

Model Invocation Methods: If a pre-classification of the data features or *model invocation* ([3], Chapter 8) occurs, then only the pre-selected model-to-data correspondences need to be considered. The pre-classification does not affect the number of nodes accepted, but it reduces the fan-out at each node and hence the number of nodes tested. The process requires an initial comparison between each model and data feature. Once in search, the search tree is the same as for the standard IT algorithm, except that at level λ, only model features known to be compatible with data feature d_λ are compared. No unary tests are needed as compatibility is ensured, but the pairwise tests still apply.

Re-ordering The Tree: This algorithm expands the IT one *model* feature at a time.

Unique Use of Features: This algorithm allows model features to be matched only once.

Visibility Subgroups: After two features match, estimate an orientation for the model, predict which model features are visible, and then expand the search tree for only these features.

Non-wildcard Search: This matching algorithm[5] explores the same search space as the standard IT, but does not use a wildcard model feature. The algorithm compared here has several new unpublished work-saving ideas: (1) members of the set Ω are generated only when needed and (2) compatible pairs of matches $(data_a, model_b)$ and $(data_c, model_d)$ are recorded to prevent being tested more than once.

Ordered Search: By exploiting an ordering of the features (e.g. size), then whenever we have successfully matched a model feature, for subsequent matches we need only consider model features after this feature in the ordering[1]. This adds an additional assumption to those used in the other algorithms.

3 The Experiments

The following simulated experimental problem is based on an example described in [8]. This allows us to compare the performance on data sets of varying sizes. (Real problems also follow below.)

Each model-match experiment consisted of: (1) initially determining a random selection of C of the D data features to be the solution and (2) for each generated model-to-data pairing, a correspondence is accepted if the new correspondence is: (a1) individually satisfied with probability p_1 and (a2) pairwise satisfied with each previously filled non-wildcard feature with probability p_2 or (b) part of the solution or use the wildcard.

For the experiments described in this paper, we used:

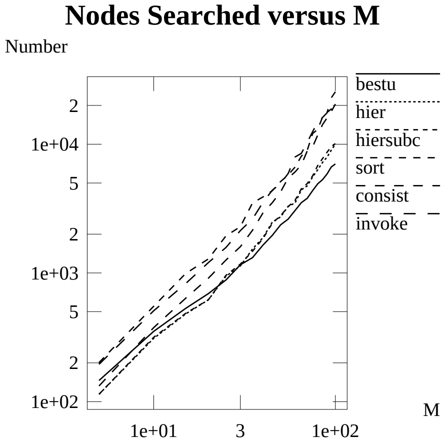
PARAMETER	NOMINAL	RANGE
M	40	5 to 100 by 5
S	20	0 to 100 by 5
p_1	0.1	0.05 to 0.75 by 0.05
p_2	0.01	0.001, 0.002, 0.004, 0.008, 0.01, 0.02 to 0.20 by 0.02, 0.25
τ	0.5	0.2 to 0.9 by 0.1
p_v	0.5	no variation
p_r	0.95	no variation

In each experiment described in this section, one parameter was varied over the range given above and all others were set to the nominal value. All experiments were run 200 times and the value reported is the mean value.

We show here only the results for varying the number of model features M and the probability p_1 (results from varying other parameters were not significantly different). Figure 1 shows how the number of nodes generated varied with the changed parameter for the best seven algorithms. In the graphs, the curves for the different algorithms are labeled by the following. The two columns at the right show the mean number of nodes generated for the maximum parameter value from the two experiments.

Label	Algorithm	$M = 100$	$p_1 = 0.75$
align	Alignment	94356	65886
geom2	Geometric+2 starters	271452	596086
geom3	Geometric+3 starters	155508	392350
geom2hash	Geometric+2 starters+indexing	50773	90416
geom3hash	Geometric+3 starters+indexing	147727	367265
hier	Hierarchy	11482	90870
hiersubc	Hierarchy+pose consistancy	12192	67983
invoke	Model invocation	25535	266019
non-WC	Non-wildcard	20487	209837
norm	Standard IT	161236	348414
reorder	Re-ordered tree	361943	348414
sort	Sorted Features	17363	62724
uniq	Unique use of feature	146563	336562
vis	Visibility subgroups	113710	195772

As we look over the results, which explore a substantial portion of the parameter spaces likely to be encountered in visual matching problems, there is no clear "winning" algorithm. The vis, geom2, geom3, norm, reorder and uniq algorithms generally have poor performance compared to the others. The real comparison is between the geom2hash, the hierarchical, the invoke, the align and the non-wildcard algorithms, and the choice depends on the problem parameters. The sorted feature algorithm also has good, but not dramatic, performance, but makes an additional problem assumption. The hier and hiersubc algorithms are generally the best when p_1 is low, and the difference between them is not large.



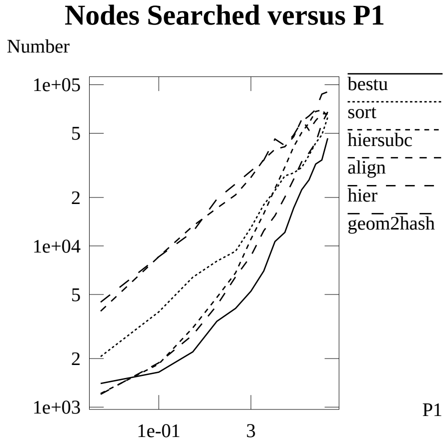


Fig. 1. Generated Nodes versus a) Number of Model Features (M) and b) Unary Match Probability (p_1). Labels are ordered by the results at the maximum parameter setting.

The non-wildcard algorithm is not bad for most problems, but its performance deteriorates when p_1 is large. The align and geom2hash algorithms become distinctly worse as M increases and the invoke algorithm becomes distinctly worse as p_1 increases.

When there is no instance of the object in the scene, the hierarchical and non-wildcard algorithms have about 6 times more search. The standard algorithm is also much worse ([8], page 389). Simulation results suggest that only the hierarchical and non-wildcard algorithms are real alternatives to the standard algorithm, and both of these algorithms give a factor of about 3–77 improvement (in search) over the standard algorithm.

To assess the performance on real data, the hierarchical and non-wildcard algorithms were compared on edge matching from several real scenes (on a SparcStation 1+, code in C++). Because the algorithms are sensitive to data feature order, the algorithms were run 100 times with the model and data features permuted randomly. The effective probabilities in this scene were $p_1 = .235$ and $p_2 = 0.017$ and the number of features were $M = 13$ and $D = 129$. Seven of 13 model edges match true data edges in the test scene. The average time taken for the matching algorithms was 0.96 sec. for the non-wildcard algorithm, 1.47 seconds for the hierarchical algorithm and 5.88 sec. for the standard algorithm. The mean number of nodes tested was 55025 for the hierarchical algorithm, 64412 for the non-wildcard algorithm and 544171 for the standard algorithm. On another test scene containing 10 instances of only the matched part, the average times required for a match was non-wildcard 20.4 sec., hierarchical 21.4 sec. and standard 419 sec. The effective probabilities in this scene were $p_1 = .288$ and

$p_2 = 0.011$ and the number of features were $M = 28$ and $D = 191$.

In a full recognition process, timings of typical associated processes are: Canny edge detector: 14.3 sec, connectivity and tracking: 2.1 sec, segmentation: 1.2 sec, merging/description: 1.9 sec and pose estimation and verification: 0.7 sec. Hence, using the improved algorithms reduces the complete time from 26 to 21 seconds in the first case and from 511 to 41 seconds in the second.

4 Discussion and Conclusions

It is obvious that the non-wildcard and hierarchical algorithms produce better performance than the more straightforward variations of the standard IT. However, for all of the algorithms, the real work occurs at the first or second step, which effectively requires a comparison between all model and data features. As any model feature might be an explanation for any data feature, it is hard to avoid this complexity, which results in MD initial comparisons and roughly $p_1 MD$ false acceptances, and which effectively provides a lower bound on the amount of work required. After that, a reduced search space is possible, but the initial effort is substantial. There does not seem to be much possibility of reducing this amount of effort, unless some additional aspect of the particular problem can be exploited.

This research was funded by SERC (IED grant GR/F/38310).

References

1. Murray, David W., Buxton, B. F. Experiments in the machine interpretation of visual motion MIT Press, Cambridge, Mass. 1990.
2. Faugeras, O. D., Hebert, M., *A 3-D Recognition and Positioning Algorithm Using Geometric Matching Between Primitive Surfaces*, Proceedings 8th Int. Joint Conf. on Artificial Intelligence, pp996-1002, 1983.
3. Fisher, R. B., From Surfaces to Objects: Computer Vision and Three Dimensional Scene Analysis, John Wiley and Sons, Chichester, 1989.
4. Fisher, R. B., *Determining Back-facing Curved Model Surfaces By Analysis At The Boundary*, Proc. 3rd Int. Conf on Computer Vision, pp 296–299, Osaka, 1990.
5. Fisher, R. B., *Non-Wildcard Matching Beats the Interpretation Tree*, Proc. 1992 British machine Vision Conf., Leeds, pp 560-569, 1992.
6. Fisher, R. B., *Hierarchical Matching Beats The Non-Wildcard and Interpretation Tree Model Matching Algorithms*, Proc. 1993 British machine Vision Conf., Surrey, pp 589–598, 1993.
7. Grimson, W. E. L., Lozano-Perez, T., *Model-Based Recognition and Localization from Sparse Range or Tactile Data*, International Journal of Robotics Research, Vol. 3, pp 3-35, 1984.
8. Grimson, W. E. L., Object Recognition By Computer: The Role of Geometric Constraints, MIT Press, 1990.
9. Huttenlocher, D. P., and Ullman, S., *Object Recognition Using Alignment*, Proc. Int. Conf. Comp. Vision, London, pp102–111, 1987.

This article was processed using the LaTeX macro package with LLNCS style