Non-Wildcard Matching Beats The Interpretation Tree

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

Abstract

Probably the best known control algorithm for high-level model matching in computer vision is the *Interpretation Tree* expansion algorithm, popularized and extended by Grimson and Lozano-Perez. This algorithm has been shown to have a high computational complexity, particularly when applied to matching problems with large numbers of features. This paper introduces a non-wildcard variation on this algorithm that has an improvement of about 4-10 in performance over the standard Interpretation Tree algorithm.

1 Introduction

Probably the most well-known control algorithm for high-level model matching in computer vision is the *Interpretation Tree*(IT) expansion algorithm, as used by Grimson and Lozano-Perez[2]. The IT algorithm searches a tree of model-to-data correspondences, such that each node in the tree represents one correspondence and the path of nodes from the current node back to the root of the tree is a set of simultaneous pairings.

Unfortunately, this algorithm has the potential for combinatorial search explosion. This has prompted researchers to develop techniques for pruning the trees, thus limiting the number of matches considered. The main techniques commonly used are based on pruning constraints[2] (which locally reject pairings that are inconsistent, and hence eliminate all of the search that might further extend this inconsistent pairing) and early termination[4]. The latter stops search (1) at the first hypothesis with a given number of pairings, or (2) at any time that it is impossible to make sufficient pairings with the remaining potential matches. However, even with these effective forms of pruning, the algorithms still can have an exponential complexity, making them unsuitable for use in scenes with many features.

As reported by Grimson[4], the main cause of the exponential complexity is the use of a "wildcard" match feature. This paper discusses and analyses a variation to the standard IT algorithm that explores a different tree without using a wildcard and requires 4-10 times less work.

2 The Standard Interpretation Tree Algorithm

Consider a set $\{ d_i \}$ of D data features and a set $\{ m_i \}$ of M model features. Then, the root of the interpretation tree has no pairings. The first level expands the root node to pair all of the M model features with data feature d_1. The second level in the tree expands each of these nodes to pair all model features with data feature d_2 (multiple pairings are allowed), and so on. The expansion continues for all D data features. At each node at level k in the tree, therefore, there is a hypothesis with k features matched.

If this IT were explored completely, there would be M^D "leaf" nodes at the bottom of the tree (i.e. these many complete interpretations) and

$$\sum_{i=0}^{D} M^i = \frac{M^{D+1} - 1}{M - 1} \doteq M^D$$

nodes in the full tree. If either M or D are of any reasonable size (e.g. larger than 5), then we can expect to have excessively large search trees.

An additional complication is that one usually wishes to include a "wild-card" model feature that will match with any data feature. This is necessary because it may not always be possible to find a model feature that matches the data feature at the current level of the tree (because of fragmentation, bad segmentation, noise, unrelated features, etc.).

One way to reduce the amount of searching is to 'prune whole branches of the tree', by showing that a given pairing or sequence of pairings is inconsistent. Therefore, all descendents from that node in the tree will also be inconsistent and need not be explored. The most common approach uses unary and binary pruning constraints. Unary constraints eliminate model-to-data pairings when some shared property is inconsistent. Binary constraints eliminate hypotheses when a relative property between a pair of model features is inconsistent with the same property between the corresponding pair of data features. For example, Grimson and Lozano-Perez[2] provide a set of *binary* constraints useful for three-dimensional scene analysis, based on *pairwise consistency constraints*, that compare quantities such as relative distance, orientation and direction. Similar constraints can be developed for higher-order consistency (e.g. vector triple products). Of particular importance is the local nature of the consistency tests, based on the assumption that a few simple, fast tests on partially generated hypotheses will eliminate large numbers of globally inconsistent hypotheses.

In the discussion below, the following quantities are used:

- there are M model features in the model.
- on average, $p_v M$ of these are visible in the scene (less than M by occlusion, being on the back side of the object, etc.). 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 and hence not visible.

Figure 1: Generated and Accepted Nodes versus Number of Model Features (M) with $S = 20$ $p_r = 0.95$ $p_1 = 0.1$ $p_2 = 0.01$ $p_v = 0.5$ $\tau = 0.5$ (loglog plot)

- of the visible model features, only p_r of these are recognizable (less than those visible because of segmentation failures, etc.) forming $C = p_r p_v M$ correct observable data features. (If the model chosen for this scene is incorrect, $p_r = 0$.) Which C of the M model features are matchable is not known initially.
- there are also S spurious data features (including noise features and visible model features that are not recognizable); hence altogether there are $D = C + S$ data features.
- the probability that a randomly chosen model feature matches with an incorrect random data feature is p_1 (correct pairings alway match).
- 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 ($\tau \in [0..1]$). Whenever this many are achieved, then the whole matching process terminates successfully immediately. Any set of matches that can never get T matches (because insufficient potential matches remain) is terminated immediately and the matching process proceeds to considering other matches.

In the discussion that follows, the term *generated* refers to nodes and paths that are created prior to testing the consistency of the node or path, and *accepted* refers to nodes or paths that pass the consistency tests.

Grimson[3] analyzed the combinatorics of the standard algorithm, and showed that, without wildcards, the algorithm tends to accept (Proposition 5, pg 274) a single path with many pairings (i.e. the correct one), and generates (Proposition 6, pg 274) a number of nodes that is quadratic in the number of model features. However, allowing a wildcard means that the algorithm will accept

Figure 2: Generated and Accepted Nodes by Spurious Features (S) with $M = 40$ $p_r = 0.95$ $p_1 = 0.1$ $p_2 = 0.01$ $p_v = 0.5$ $\tau = 0.5$ (loglog plot)

an exponential number of correctly matchable features. One key term is 2^C, arising from the power set of the C matchable features. The complexity occurs because each matchable data feature can be either matched with the correct model feature or the wildcard. Examination of a typical search tree shows that most of the tree consists of paths containing either members of this power set or wildcards. Many of these paths can be eliminated by using the *termination threshold* described above. This can only apply when the search is sufficiently advanced, but it does make a significant improvement.

Grimson[3] analyzed the consequences of this termination condition and showed (Corollary 3.2, pg 367) that if:

$$p_2 MD < 2$$

then the expected number of nodes generated is bounded by:

$$\frac{MD^2}{C} < num\_generated < aT\frac{MD^2}{C}$$

where a is a small constant. There might be some problems with the exactness these bounds, but the conclusion that the use of a termination condition improves performance is valid.

Unfortunately, the $p_2 MD < 2$ condition given above does not always hold, in which case the algorithm again seems to be exponential. In fact, in the experiments described below, it only holds for the smallest test cases.

3 The Non-wildcard Matching Algorithm

The vast number of nodes in the standard algorithm arises because of the use of wildcards. An alternative search algorithm explores the same search space, but

Figure 3: Generated and Accepted Nodes by Unary Match Probability (p_1) with $M = 40$ $S = 20$ $p_r = 0.95$ $p_2 = 0.01$ $p_v = 0.5$ $\tau = 0.5$ (loglog plot)

does not use a wildcard model feature matched to data features. The essence of the difference is the search process skips over all data pairings that use a wildcard, to consider the next true data-model feature pairing. This results in a flattening of the search tree. The algorithm has two phases:

1. The set $\Omega = \{s_k\} = \{(m_{i(k)}, d_{j(k)})\}, k = 1..N$ of all pairs of features satisfying the unary pairing constraints is formed, such that if s_r is before s_s (i.e. $r < s$), then $j(r) \leq j(s)$.

2. A different search tree is explored, in which each extension of a branch is formed by appending new entries from Ω, subject to the constraints that (1) each data feature appears at most once on a path through the tree and (2) the data features are used in order (with gaps allowed).

Starting from a branch ending with pair s_λ (or nothing at the root of the tree), all pairs $s_{\lambda+1} \ldots s_N$ are possible extensions to the branch. Only extensions that satisfy the normal binary constraints are accepted. Extension stops when the termination number of matches is reached, or on branches where insufficient possibilities remain in the tail of Ω.

For example, if $\Omega = \{s_1, s_2, s_3, s_4\} = \{(m_2, d_1), (m_4, d_2), (m_1, d_2), (m_5, d_4)\}$, the tree:

			X		
	s_1		s_2	s_3	s_4
s_2	s_3	s_4	s_4	s_4	
s_4	s_4				

is searched depth first following the leftmost branches first (no pruning is shown here to illustrate the shape of the tree). The initial step considers the individual model-data pairings once (i.e. the unary constraints are tested once instead of whenever needed, as in the IT tree). As the second and third levels of the new search tree contain complete matches, the binary constraints eliminate almost all false pairings quickly. The trade-off is that the branching factor of the new tree is $sizeof(\Omega)$ instead of M. This search algorithm can produce the same set of hypotheses as the standard IT algorithm, with respect to the data features paired to non-wildcard model features. The order of generation may be different when the termination threshold is used.

4 The Experiments

To demonstrate the effectiveness of the non-wildcard search algorithms, we use the following experimental problem. The approach is designed to allow comparison of methods for which no formal complexity measure has yet been determined, and also to allow comparison of algorithms within the same complexity class. The problem is based on an example described in Grimson[4]. The experiments use simulated data. However, Grimson showed that the model and simulation gave a reasonable characterization of real matching problems. The use of the simulated problems then allows us to compare the algorithm performance on the same data sets of varying sizes.

Based on the problem model given in Section 2, each model-match experiment of the two algorithms will consist of:

1. Initially determining a random selection of C of the D data features to be the solution.

2. For each generated model-to-data pairing, a correspondence that is not part of the solution and does not use a wildcard is accepted if the new correspondence is individually satisfied with probability p_1 and the new correspondence is pairwise satisfied with each previously filled non-wildcard feature with probability p_2. Correspondences that are part of the solution or use the wildcard are accepted.

The experiments with the non-wildcard search tree algorithm are similar. 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, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20
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.

Figure 4: Generated and Accepted Nodes by Binary Match Probability (p_2) with $M = 40$ $S = 20$ $p_r = 0.95$ $p_1 = 0.1$ $p_v = 0.5$ $\tau = 0.5$ (loglog plot)

The graphs in Figures 1–5 given show how the number of nodes generated and accepted varied with the parameters for the new and standard IT algorithms.

As we look over the results, which explore a substantial portion of the parameter spaces likely to be encountered in visual matching problems, we can see that the non-wildcard is clearly better than the standard IT algorithm. In search, the non-wildcard algorithm is not bad for most problems, but its performance deteriorates as p_1 increases (this increases the number of possible matches to consider at each stage). For acceptances, the non-wildcard algorithm is also the better, as it does not allow proliferating wildcard hypotheses. Except when p_1 is large, the non-wildcard algorithm did about 4 times less search and 10 times less accepting than the standard algorithm.

One might also consider how the two algorithms perform when there is no instance of the object in the scene. Then, it is unlikely that the early success conditions would occur, and thus almost all of the search space would have to be explored. Figure 6 shows the number of nodes generated and accepted in this case. When there is no true match possible, the non-wildcard is still much better, but in both cases much more work is done (e.g. about 10-30 times more work). Grimson ([3], page 389) shows that the standard algorithm is also much worse when no match is possible.

5 Computational Complexity of the Non-Wildcard Matching Algorithm

Grimson[3] has mainly concentrated on estimating upper and lower bounds for the standard algorithm. As seen in the results from the previous section, the non-wildcard search algorithm looks very promising. Hence, we give here the result of a complexity analysis for that algorithm, except that we state here

Figure 5: Generated and Accepted Nodes by Acceptance Threshold (τ) with $M = 40$ $S = 20$ $p_r = 0.95$ $p_1 = 0.1$ $p_2 = 0.01$ $p_v = 0.5$ (loglog plot)

(without proof) the *mean* performance of the algorithm.

Theorem 1 (Mean Complexity of Non-Wildcard Algorithm) *Given the problem definitions from above, there are expected to be C true pairings and $F = p_1(MD - C)$ false pairings that arise from the initial model to data feature matching. Assume that M and D are very large, so that the effect of matching one feature does not significantly affect the rest of the algorithm. Also assume that no false hypotheses containing 3 or more pairings survive the pruning tests (i.e. $Fp_2 < 1$). Then, the expected amount of search is approximately:*

$$MD + T + \frac{F}{C}(C+F) + p_2\frac{F}{C}(C+F-T)(C+F-T+1)) \doteq O(M^5)$$

and the expected number of hypotheses accepted is approximately:

$$T + \frac{F}{C} + p_2\frac{F}{C}(C+F-T+1) \doteq O(M^3)$$

6 Discussion and Conclusions

As Grimson observed, most of the complexity of the standard interpretation tree search is a consequence of the use of "wildcards" to overcome missing and erroneous data. However, merely having "good" data does not mean one can avoid the use of the wildcard, because the so-called false features may have arisen from other objects in the scene, or other subcomponents of the object being recognized. Hence, the wildcard is likely to remain a key element of the general interpretation tree search algorithm. If one could assume that there were only a limited amount of scene clutter, then one might limit the use of wildcards to a specific number. However, more than one or two would still allow a considerable number of partially empty hypotheses.

Figure 6: Generated and Accepted Nodes versus Number of Model Features (M) When No Instance of the Model is Present with $S = 20$ $p_r = 0.95$ $p_1 = 0.1$ $p_2 = 0.01$ $p_v = 0.5$ $\tau = 0.5$ (loglog plot)

From the experiments, it is obvious that the non-wildcard algorithm produces better performance than the standard IT matching algorithm. For the non-wildcard algorithm, 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, which provides a lower bound on the amount of work required. After that, a reduced search space needs to be considered, 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.

Real benefits can be gained by reducing the number of features that need to be considered at a time. If the data features can be partitioned into K subsets, which can be matched independently, and the models features can also be partitioned into L corresponding subcomponents, then the brute-force version of the matching algorithm is reduced from M^D to:

$$KL(\frac{M}{L})^{\frac{D}{K}}$$

which is considerably less. This requires perceptual organization [5], such as a region or surface patch grouping (e.g. [1] Chapter 5).

The analysis above also assumed that only one model (i.e. one set of model features) was considered for matching. If all models must be considered, then the computational complexity will be high, as the results in Section 4 showed. Hence, some form of model invocation method is needed to reduce the number of candidate models (e.g. [1] Chapter 8, [3] Chapter 15).

The net conclusion is that by using the non-wildcard algorithm as an alternative to the standard interpretation tree visual matching algorithm, it is possible to reduce the amount of search by a factor of about 4 and number of partial interpretations accepted by a factor of 10, where the precise amount of improvement depends on the problem parameters. Both factors are important, because, depending on the particular matching algorithm, the savings achieved depend on relative costs of each action (e.g. the pairwise consistency checking costs may high relative to final verification costs).

The relative speed difference of the implemented matching algorithms might overcome this reduction in theoretical search complexity. However, the $M = 100$ case from Figure 1, matching requires 1.27 seconds for the non-wildcard algorithm, as compared to 5.4 seconds for the standard algorithm (on a Sparc-Station 1+). Hence, the speed of the non-wildcard algorithm is also significantly better than the standard algorithm in the implementations compared.

Acknowledgements

This research was funded by SERC (IED grant GR/F/38310). Other facilities provided by University of Edinburgh. This paper benefited greatly from discussions with Dibio Borges, John Hallam, Howard Hughes, Mark Orr, Kristian Simsarian, Manuel Trucco and Mike Uschold.

References

[1] Fisher, R. B., From Surfaces to Objects: Computer Vision and Three Dimensional Scene Analysis, John Wiley and Sons, Chichester, 1989.

[2] 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.

[3] Grimson, W. E. L., Object Recognition By Computer: The Role of Geometric Constraints, MIT Press, 1990.

[4] Grimson, W. E. L., *The Combinatorics of Heuristic Search Termination for Object Recognition in Cluttered Environments*, Lecture Notes in Computer Science, ECCV-90, Springer-Verlag, pp 552-556, 1990.

[5] Witkin, A. P., Tenenbaum, J. M., *What Is Perceptual Organization For?*, Proceedings 8th IJCAI, pp1023-1026, 1983.

A Non-Wildcard Search Algorithm

```
// Non-wildcard expansion variation on standard algorithm:
//   expand tree by members of valid_pairs (not by data levels),
//   subject to not reusing data features.
searchtree(treesofar, valid_pairs)
{   if empty(valid_pairs) return fail
```

```
    trylist = valid_pairs
    do {
        if can never get enough return fail
        extension = head(trylist)
        trylist = tail(trylist)
        if data feature in extension already appears in treesofar
            then skip this extension
        if compatible(extension, treesofar)
        {   if enough matches return success
            if success(searchtree(append(treesofar,extension),
                trylist)), then return success}
    } while non-empty(trylist)
    return fail}

// test for compatibility of new pairing with rest of pairings:
boolean compatible(new_pair, treesofar)
{   // check pairwise with previously filled slots of this hyp
    for each pair in treesofar
        if not compatible2(pair, new_pair) then return false
    return true}
```