A Large Deviation Theory Analysis of Bayesian Tree Search

J.M. Coughlan
Smith-Kettlewell Eye Research Institute
San Francisco, CA 94115

A.L. Yuille
Department of Statistics
University of California at Los Angeles
Los Angeles, CA 90095
yuille@stat.ucla.edu

In Mathematical Methods in Computer Vision. Eds. P. Olver and A. Tannenbaum. IMA Volumes in Mathematics and its Applications, Vol. 133, pp 1-17. Springer-Verlag, New York Inc. 2003.

A LARGE DEVIATION THEORY ANALYSIS OF BAYESIAN TREE SEARCH

JAMES M. COUGHLAN* AND ALAN L. YUILLE*

Abstract. Many perception, reasoning, and learning problems can be expressed as Bayesian inference. We point out that formulating a problem as Bayesian inference implies specifying a probability distribution on the ensemble of problem instances. This ensemble can be used for analyzing the expected complexity of algorithms and also the algorithm-independent limits of inference. We illustrate this by analyzing the problem of road detection, formulated as tree search, by Geman and Jedynak [6]. This analysis uses large deviation theory to put bounds on the probability of rare events, such as exploring wrong branches of the tree in depth. We prove that the expected convergence is linear in the size of the road (i.e. depth of the tree) even though the worst case performance is exponential.

Key words. Bayesian inference, large deviation theory, tree search.

1. Introduction. Many problems in vision, speech, reasoning, and other sensory and control modalities can be formulated as Bayesian inference [9]. It is important to understand the complexities of algorithms which can perform these inferences.

We point out that formulating a problem as Bayesian inference implies specifying a probability distribution on the *ensemble of problem instances*. More formally, in Bayesian inference the goal is to estimate a quantity x from data y by using the *posterior* distribution $P(x|y)$. Constructing this posterior requires specifying a *likelihood function* $P(y|x)$ and a *prior* distribution $P(x)$. From these distributions we can construct a distribution $P(x, y)$ on the ensemble of problem instances (x, y). See Figure 4 for samples from a particular ensemble for road tracking.

There are two main advantages to analyzing the performance of an algorithm over the ensemble of problem instances. Firstly, it allows us to determine the behavior of the algorithm for typical problem instances (i.e. those which occur with non-negligible probability) and means that we may not have to deal with worst case situations (because they may have arbitrarily small probabilities). Secondly, having a distribution over the ensemble of problem instances also enables us to quantify the accuracy of the estimates found by the algorithm.

In this paper, we illustrate these advantages by analyzing the complexity of an algorithm proposed by Geman and Jedynak [6] for detecting roads in aerial images. Geman and Jedynak formulated the problem as Bayesian maximum a posteriori (MAP) estimation. This reduces the problem to tree search. In this paper we analyze the complexity of a variant of the Geman and Jedynak algorithm (this variant was proposed by the authors in [18]).

*Smith-Kettlewell Eye Research Institute, 2318 Fillmore St., San Francisco, CA 94115.

The complexity, and performance, of the algorithm depends on the probability distributions which characterize the ensemble of problem instances (i.e. the likelihood function and the prior). For a large class of ensembles, we are able to prove that the expected complexity is linear in the length N of the road (by contrast, the worst case complexity is exponential in N). We are also able to put bounds on the expected errors made by the algorithm.

We emphasize that we are concerned with the ability of the algorithm to detect the road path *which may not necessarily correspond to the MAP estimate*. For any problem instance there are three important paths: (i) the true road path, (ii) the MAP estimate of the true road path, and (iii) the path found by the algorithm. In this paper we are concerned with the difference between (iii) and (i) only.

These results complement our previous work [20, 21] which used this ensemble concept to analyze the *algorithm-independent* performance of MAP estimation on problems of this type (i.e. we evaluated errors between the MAP estimate (ii) and the true road path (i)). In particular, we derived an *order parameter* K_B which is a function of the ensemble. We proved that if $K_B < 0$ then it is impossible to detect the true road by *any algorithm*. (Not surprisingly, our results in this paper on linear expected complexity only apply to ensembles for which $K_B > 0$).

Another advantage of the ensemble concept is illustrated by our choice of a *heuristic* A^* algorithm [18]. A* algorithms [11, 16, 13] search trees and/or graphs using a *heuristic* to estimate future rewards. If we are using a Bayesian ensemble then the probability distributions can be used to generate heuristics.

Technically, our proofs make use of *large deviation theory* [7]. In particular we use Sanov's theorem, see [4], to put bounds on the probability of rare events. We note that many results on statistical learning theory [15] are derived using similar techniques from large deviation theory.

Finally, we would like to mention related work on optimization which uses the concept of ensembles.

Firstly, Karp and Pearl [8, 11] provided a theoretical analysis of convergence rates of A* search by considering an ensemble of problem instances. They studied a binary tree where the rewards for each arc were 0 or 1 and were specified by a probability p. They then obtained the complexity of algorithms for finding the minimum cost path. This work has some similarities to ours but their formulation is not Bayesian, their heuristics for A* algorithms are different, and large deviation theory is not used. Their work was an inspiration for us and we provided an analysis of a block pruning algorithm motivated by them in [17].

Secondly, there are some recent studies showing that order parameters exist for NP-complete problems and that these problems can be easy to solve for certain values of the order parameters [1, 14]. This work involves analyzing ensembles of problem instances. But the distribution of instances

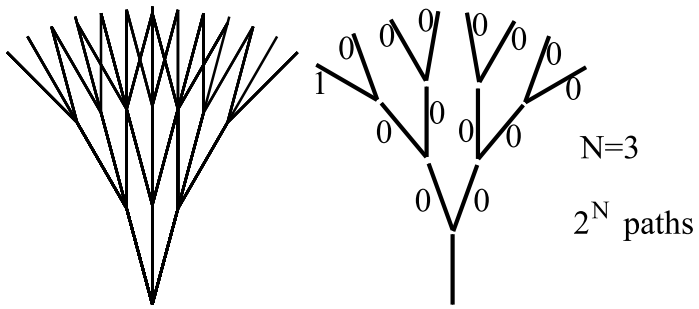

FIG. 1. *Left Panel: Geman and Jedynak's tree structure with a branching factor of* $Q = 3$. *Right Panel: The worst case complexity is exponential because, for some problem instances, the best path is determined only by the reward of the final arc segment. In this panel* $Q = 2$ *and all the* 2^N *paths have to be examined.*

in the ensemble is typically assumed to be uniform and is not derived from Bayesian methods.

The structure of this paper is as follows. In Section (2) we formulate the road tracking problem and introduce A*. Section (3) gives complexity results for a special choice of A* heuristic (making use of Sanov's theorem). These results can be extended to other heuristics [2]. Section (4) proves that sorting the queue for A* takes constant expected time per sort operation.

2. Problem formulation.

2.1. Tree search: the Geman and Jedynak model. Many problems in artificial intelligence can be formulated as tree search ([16, 11, 13]). We now study a specific example of this class of problem.

Geman and Jedynak [6] formulate road detection as tree search in a Q-nary tree, see Figure 1. The starting point and initial direction are specified and there are Q^N possible distinct paths down the tree. The goal is to find the road path (whose statistical properties differ from those of the non-road paths). The worst case complexity for this problem is exponential but, as we will prove, in many circumstances it is possible to detect a good approximation to the road path in linear expected time. Our analysis involves considering an ensemble of problem instances.

More formally, a road hypothesis, or path, consists of a set of connected straight-line *segments*. We can represent a path by a sequence of moves $\{t_i\}$ on the tree. Each move t_i belongs to an *alphabet* $\{b_\nu\}$ of size Q. For example, the simplest case studied by Geman and Jedynak sets $Q = 3$ with an alphabet b_1, b_2, b_3 corresponding to the decisions: (i) b_1 – go straight (0 degrees), (ii) b_2 – go left (-5 degrees), or (iii) b_3 – go right (+ 5 degrees).

Each tree will contain a *target path* which corresponds to the road to be detected. This path is sampled from a prior probability distribution $P(\{t_i\}) = \prod_{i=1}^{N} P_{\Delta G}(t_i)$, where $P_{\Delta G}(.)$ is the geometric transition proba-

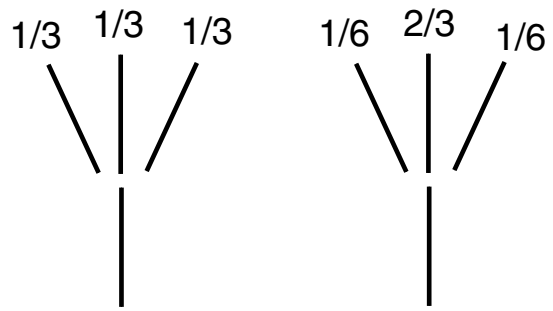

FIG. 2. *Different priors for the geometry. (Left Panel) the probabilities of turning left, right, or straight are* $1/3$. *(Right Panel) the probability of going straight is* $2/3$ *and the probabilities of turning right or left are* $1/6$ *each, biasing towards straighter paths.*

bility. For our $Q = 3$ example, we may choose to go straight, left or right with equal probability (i.e. $P_{\Delta G}(b_1) = P_{\Delta G}(b_2) = P_{\Delta G}(b_3) = 1/3$), see Figure 2.

A sequence of moves $\{t_i : i = 1, ..., N\}$ determines a path of segments $X = (x_1, ..., x_N)$ (ie. these segments form a connected path from the top of the tree to the bottom). Conversely, a consistent path X determines a sequence of moves. (This also applies to subpaths). Let χ denote all the segments of the tree. So a path X is a connected subset of χ. The set of segments not on the path is the complement $\chi \setminus X$.

There is an *observation* y_x for each segment $x \in \chi$ of the tree. The set of all observations is $Y = \{y_x : x \in \chi\}$. The values of the observations belong to an alphabet $\{a_\mu\}$ of size J. An observation y_x is drawn from a distribution $P_{on}(.)$ if the segment is on the target path (ie. if $x \in X$). If not, it is drawn from $P_{off}(.)$. For any path $\{t_i\}$ through the tree, with segments $\{x_i\}$, we have a corresponding set of observations $\{y_{x_i}\}$. (The observation y_x is the response to a non-linear filter, evaluated on the image at segment x, which is designed to detect straight road segments. The filter is trained on examples of on-road and off-road segments to determine empirical distributions $P_{on}(y)$ and $P_{off}(y)$, as described in [6]. See Figure 3 for examples of distributions P_{on}, P_{off}, taken from [10]).

This determines the likelihood function $P(Y|X)$:

$$P(Y|X) = \prod_{x \in X} P_{on}(y_x) \prod_{x \in \chi \setminus X} P_{off}(y_x), \tag{2.1}$$

which we can re-express as:

$$P(Y|X) = \prod_{i=1,...,N} \frac{P_{on}(y_{x_i})}{P_{off}(y_{x_i})} \prod_{x \in \chi} P_{off}(y_x) = \prod_{i=1,...,N} \frac{P_{on}(y_{x_i})}{P_{off}(y_{x_i})} F(Y). \tag{2.2}$$

where $F(Y) = \prod_{x \in \chi} P_{off}(y_x)$ is independent of the target path X.

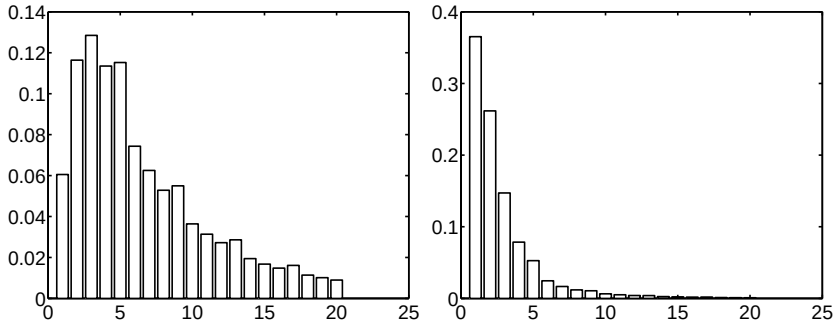

FIG. 3. *The quantized distributions $P_{on}(y)$ (Left) and $P_{off}(y)$ (Right), where $y = \left|\vec{\nabla} I(\mathbf{x})\right|$, learned from image data. Observe that, not surprisingly, $\left|\vec{\nabla} I(\mathbf{x})\right|$ is likely to take larger values on an edge rather than off an edge.*

We formulate the problem as MAP estimation to find the mode of the posterior distribution $P(X|Y) = P(Y|X)P(X)/P(Y)$ where $P(X) = \prod_{i=1}^{N} P_{\Delta G}(t_i)$ (where $\{t_i\}$ is the sequence of moves that generates the path X).

Then MAP estimation for X is equivalent to maximizing

$$P(Y|X)P(X) = \prod_{i=1}^{N} P_{\Delta G}(t_i) \prod_{i=1,...,N} \frac{P_{on}(y_{x_i})}{P_{off}(y_{x_i})} F(Y). \tag{2.3}$$

This is equivalent to finding the path $\{t_i\}$ with observations $\{y_i\}$ which maximizes the following *reward function*:

$$r(\{t_i\},\{y_i\}) = \sum_{i=1}^{N} \log\left\{\frac{P_{on}(y_i)}{P_{off}(y_i)}\right\} + \sum_{i=1}^{N} \log\left\{\frac{P_{\Delta G}(t_i)}{U(t_i)}\right\}, \tag{2.4}$$

where y_i is shorthand for y_{x_i}, $U(.)$ is the uniform distribution (i.e. $U(b_\nu) = 1/Q \ \forall \nu$) and so $\sum_{i=1}^{N} \log U(t_i) = -N \log Q$ which is a constant. The introduction of $U(.)$ helps simplify the analysis in the following subsections.

Observe that the reward of a particular path depends only on the variables $\{t_i\},\{y_i\}$ which define the path (the moves and the observations). This is because the factor $F(Y)$ in $P(Y|X)$ is independent of X and can be ignored (i.e. it does not affect which path is most probable).

For any path (or subpath) in the tree of length n we can re-express the reward function $r(\{t_i\},\{y_i\})$ as:

$$r(\{t_i\},\{y_i\}) = n\vec{\phi} \cdot \vec{\alpha} + n\vec{\psi} \cdot \vec{\beta}, \tag{2.5}$$

where $\vec{\alpha}$ and $\vec{\beta}$ have components:

$$\alpha_\mu = \log \frac{P_{on}(a_\mu)}{P_{off}(a_\mu)}, \ \mu = 1,...,J, \quad \beta_\nu = \log \frac{P_{\Delta G}(b_\nu)}{U(b_\nu)}, \ \nu = 1,...,Q. \tag{2.6}$$

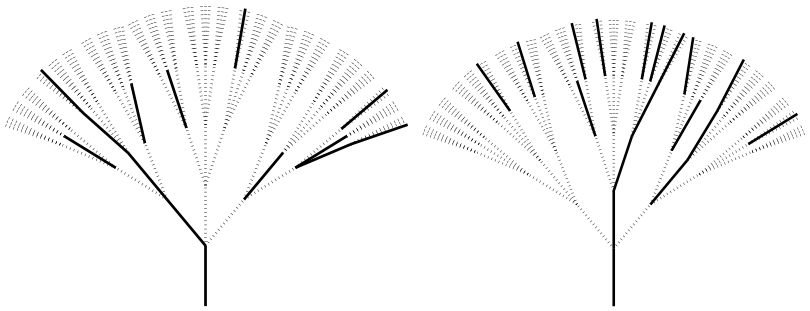

FIG. 4. *Samples from the Bayesian ensemble. Simulated road tracking problem where dark lines indicate strong edge responses and dashed lines specify weak responses. The data was generated by stochastic sampling using a simplified version of the models analyzed in this paper. In both examples there is only one strong candidate for the best path (the continuous dark line) but chance fluctuations have created subpaths in the noise with strong edge responses.*

and $\vec{\phi}$ and $\vec{\psi}$ are normalized histograms, or *types* [4], with components (with $\delta_{i,j}$ denoting the Kronecker delta function):

$$\phi_\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{y_i,a_\mu},\ \mu = 1,...,J\ ,\ \psi_\nu = \frac{1}{n}\sum_{i=1}^{n}\delta_{t_i,b_\nu}\ \nu = 1,...,Q. \tag{2.7}$$

We now illustrate the Bayesian ensemble by Figure 4 which consists of two samples from the ensemble. In these cases the target path is easily detectable (i.e. the target reward is higher than the reward of any other path) but noise fluctuations mean that some subpaths may distract the algorithm from the target. In other ensembles, the target path may be far harder to detect.

2.2. Can the task be solved? Distractor paths. The goal is to detect the target path by selecting the path with highest reward. But the MAP estimate may not necessarily correspond to the target path. In this subsection, we specify conditions which ensure that the MAP estimate is expected to be significantly *similar* to the target path. Unless these conditions are satisfied it will be *impossible to find the target path by any algorithm.*

The tree contains one target path and $Q^N - 1$ distractor paths. We categorize the distractor paths by the stage at which they diverge from the target path, see Figure 5. For example, at the first branch point the target path lies on only one of the Q branches and there are $Q-1$ false branches which generate the first set of false paths F_1. Now consider all the $Q-1$ false branches at the second target branch, these generate set F_2. As we follow along the true path we generate sets F_i of size $(Q-1)Q^{N-i}$. The set of all paths is therefore the target path plus the union of the F_i $(i = 1,\ldots,N)$.

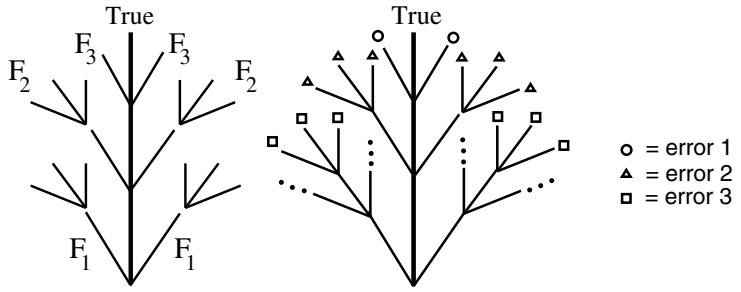

FIG. 5. *Left: Given a specified target path (the straight line shown in bold in this case) we can divide the set of paths up into N subsets $F_1, ..., F_N$ as shown here. Paths in F_1 have no overlap with the target path. Paths in F_2 overlap with one segment, and so on. Intuitively, we can think of this as an onion where we peel off paths stage by stage. Right: When paths leave the target path they make errors which we characterize by the number of false segments. For example, a path in F_1 has error N, a path in F_i has error $N + 1 - i$.*

To determine whether the target path can be found by MAP estimation we must consider the probability that one of the distractor paths has higher reward than the target path. For example, consider the probability distribution $\hat{P}_{1,max}(r_{max}/N)$ of the *maximum* reward (normalized by N) of all the paths in F_1. We can compare this to the probability distribution of the (normalized) reward $\hat{P}_T(r_T/N)$ of the target path. In related work [21], we use techniques similar to Sanov's theorem to estimate these quantities and to show that there is a phase transition depending on a parameter K given by:

$$K = D(P_{on}||P_{off}) + D(P_{\Delta G}||U) - \log Q, \tag{2.8}$$

where $D(P_{on}|P_{off}) = \sum_y P_{on}(y) \log \frac{P_{on}(y)}{P_{off}(y)}$ is the Kullback-Leibler divergence between P_{on} and P_{off}.

If $K > 0$ then the probability distribution for the target path reward $\hat{P}_T(r_T/N)$ lies to the right of the distribution $\hat{P}_{1,max}(r_{max}/N)$ of the maximum reward of paths in F_1, see Figure 6 left panel, and it is straightforward to detect the target path. At $K \approx 0$ the two distributions overlap and it becomes hard to detect the target path, see Figure 6 center panel. But if $K < 0$, then $\hat{P}_{1,max}(r_{max}/N)$ is to the right of $\hat{P}_T(r_T/N)$, see Figure 6, and it is impossible to detect the target path.

To get intuition for K we consider its three terms. The first term, $D(P_{on}||P_{off})$, is a measure of how effective the local filter cues are for detecting the target. If $P_{on} = P_{off}$ then $D(P_{on}||P_{off}) = 0$ and the local cues are useless. The second term $D(P_{\Delta G}||U)$ is a measure of how much prior knowledge we have about the probable shape of the target (setting $P_{\Delta G} = U$ means we have no prior information). Finally, $\log Q$ is a measure of how many distractor paths there are. Therefore K becomes larger (and

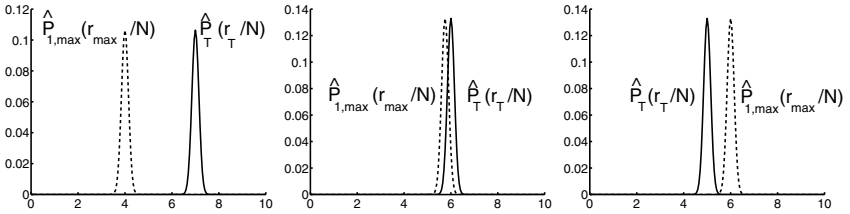

FIG. 6. *A schematic illustration of the Phase Transition. (Top left panel) The reward of the target path is higher than the largest reward of the distractor paths (so detection is straightforward). (Top right panel) The task becomes difficult because the target reward and the best distractor reward are very similar. (Bottom panel) Detecting the target path becomes impossible because its reward is lower than the best distractor path. The horizontal axis labels the normalized reward.*

so target detection becomes easier) with better filter detectors, more prior knowledge, and fewer distractor paths.

In related work [20] we used Sanov's theorem to show that the expected number of paths in F_1 with rewards greater than the target path behaves as 2^{-NK_B}, where the *order parameter* K_B is defined by:

$$K_B = 2B(P_{on}, P_{off}) + 2B(P_{\Delta G}, U) - \log Q, \tag{2.9}$$

where $B(P,Q) = -\log \sum_{i=1}^{m} (p_i)^{1/2}(q_i)^{1/2}$ is the Bhattacharyya bound between the distributions $P = \{p_i\}$ and $Q = \{q_i\}$. Once again, there is a change in behavior as K_B changes sign. For $K_B > 0$, we expect there to be no paths in F_1 with rewards greater than the target path.

Our analysis of the A* algorithm will proceed in the regime where $K > 0$ and $K_B > 0$. There is little purpose in estimating how fast one can compute the MAP estimator unless one is sure that the estimator is detecting a good approximation to the correct target. Our results, see section (3), will require an additional condition to hold, see Theorems 6 and 7, which will ensure that $K_B > 0$. There will, however, be situations where the target is detectable (ie. $K_B > 0$) but where we cannot prove expected linear convergence. More specifically, we can express $K_B = \{\psi_1 - \log Q\} + \psi_2$ where ψ_1, ψ_2 are positive quantities which are defined in Theorem 3. Our complexity proofs apply provided $\psi_1 > \log Q$. If $\psi_1 < \log Q$ but $\psi_1 + \psi_2 > \log Q$ then the target path is detectable but we can say nothing about the complexity of the algorithms.

2.3. A*. The A* graph search algorithm [11, 16, 13] is used to find a path of maximum reward between a start node A and a goal node B in a graph, see Figure 7. The reward of a particular path is the sum of the rewards of each edge traversed. The A* procedure maintains a tree of partial paths already explored, and computes a measure f of the "promise" of each partial path (i.e. leaf in the search tree). New paths are considered by extending the most promising node one step. The measure

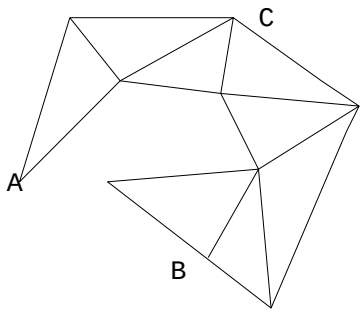

FIG. 7. *The A* algorithm tries to find the path from A to B with highest reward. For a partial path AC the algorithm stores $g(C)$, the best reward to go from A to C, and an overestimate $h(C)$ of the reward to go from C to B.*

f for any node C is defined as $f(C) = g(C) + h(C)$, where $g(C)$ is the best cumulative reward found so far from A to C and $h(C)$ is an overestimate of the remaining reward from C to B. The closer this overestimate is to the true reward then the faster the algorithm will run. We will refer to the value of f as the *A* reward* in contrast with the reward function of equation (2.5).

It is straightforward to prove that A* is guaranteed to converge to the correct result provided the heuristic $h(.)$ is an upper bound for the true reward from all nodes C to the goal node B. A heuristic satisfying these conditions is called *admissible*. Conversely, a heuristic which does not satisfy them is called *inadmissible*. The word "inadmissible" is a technical term only and *does not* imply that inadmissible heuristics are inferior to admissible ones. In fact, as we show in this paper, algorithms using inadmissible heuristics can converge rapidly to good approximations to the correct result. Conversely, as discussed below, algorithms with admissible heuristics may be slow to converge.

In this paper, we consider inadmissible heuristics. We set the heuristic reward to be $H_L + H_P$ for each unexplored segment (where $_L$ and $_P$ label the likelihood and the geometric prior respectively). Thus a subpath starting at the origin of length M will have heuristic reward of $(N - M)(H_L + H_P)$. We will drop the $N(H_L + H_P)$ term, which is the same for all paths, and simply use $-M(H_L + H_P)$ as the heuristic.

We consider the A* rewards of two partial paths, one of length m segments that overlaps completely with the target path, and the other of length n that does not overlap at all with the target path. The A* rewards of these paths are denoted by $S_{on}(m)$ and $S_{off}(n)$ and are given by:

$$
\begin{aligned}
S_{on}(m) &= m\{\vec{\phi}^{on} \cdot \vec{\alpha} - H_L\} + m\{\vec{\psi}^{on} \cdot \vec{\beta} - H_P\}, \\
S_{off}(n) &= n\{\vec{\phi}^{off} \cdot \vec{\alpha} - H_L\} + n\{\vec{\psi}^{off} \cdot \vec{\beta} - H_P\},
\end{aligned}
\tag{2.10}
$$

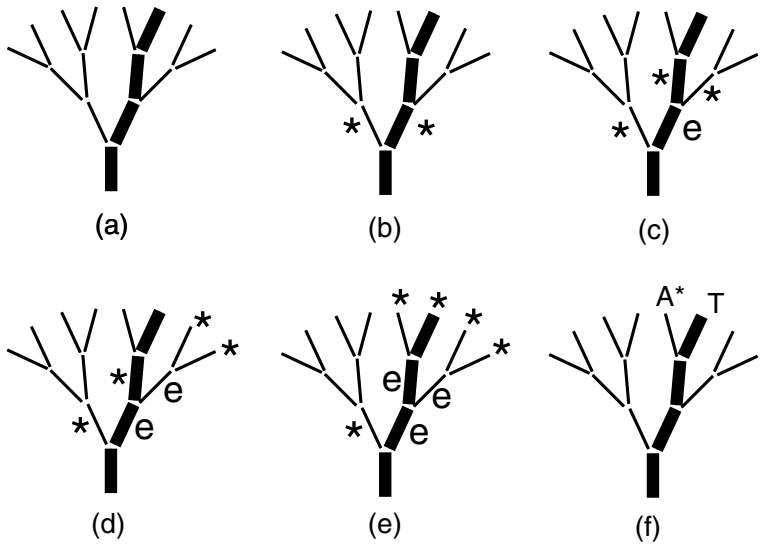

FIG. 8. *An A* search sequence. Panel (a) shows the target path in bold. Panel (b) shows the two segments added to the queue (marked by asterisks) at the start of the search. Panel (c) explores the right segment, removes it from the queue (label e), and adds its two children to the queue. The search continues in panels (d,e) where explored segments are eliminated from the queue (and labelled e) and their children are added to the queue (labelled with asterisks). Panel (f) A* converges to a solution (A*) that is one segment away from the target path (T).*

where $\phi_\mu^{on} = \frac{1}{m}\sum_i \delta_{y_i,a_\mu}$ and $\psi_\nu^{on} = \frac{1}{m}\sum_i \delta_{t_i,b_\nu}$ for the on path segments (and similarly for the off path segments).

The effect of this heuristic is to encourage us to explore subpaths provided their reward per segment is greater than $H_L + H_P$. Note that the larger the value of $H_L + H_P$ the more the algorithm will favor a breadth-first strategy [16] of exploring the tree (because few long subpaths will have rewards per segment exceeding $H_L + H_P$). Breadth-first search is a conservative strategy which will find the best solution but may take a long time to do so. In general, the smaller the heuristic then the faster the search time but the greater the possibility of error. In particular, if $H_L + H_P > \max_{\mu\in\{1,..,J\}} \alpha_\mu + \max_{\nu\in\{1,...,Q\}} \beta_\nu$ then the heuristic is admissible and the search is guaranteed to converge to the path with highest reward [11, 16, 13]. (But an algorithm which uses this heuristic is likely to be slow.) We will be considering inadmissible heuristics (i.e. $H_L + H_P < \max_{\mu\in\{1,..,J\}} \alpha_\mu + \max_{\nu\in\{1,...,Q\}} \beta_\nu$) which we expect to be quicker than admissible heuristics but which will have more errors (our theorems quantify these statements).

There is a close connection between heuristics for A* search and the general issue of pruning search algorithms. In previous work [19] we analyzed the results of a search algorithm which pruned out paths whose

rewards fell below a critical value (corresponding to a heuristic). In related work, we explored the use of pruning heuristics for speeding up dynamic programming algorithms for detecting hand outlines in real images [3].

We will first prove convergence results for a specific choice of H_L, H_P and later generalize to a larger set of values.

3. Convergence proof with Bhattacharyya Heuristics. This section will prove convergence results for a specific choice of heuristic which we call the *Bhattacharyya Heuristic*. The next section will generalize the results to a larger class (for which the proofs are more complicated). First we need to say something about the choice of heuristics.

We need to choose a heuristic so that it is smaller than the reward (per unit length) that we would get from the target path and larger than the reward for a distractor path. This means, see equation (2.10), that the A* rewards will tend to be positive for the target and negative for the distractor paths (so the algorithm will prefer to explore the target). Let us consider the reward H_L only (the analysis is similar for H_P). The expected reward for the target path is $D(P_{on}||P_{off}) = \sum_y P_{on}(y) \log \frac{P_{on}(y)}{P_{off}(y)}$ and for the distractor path it is $-D(P_{off}||P_{on}) = \sum_y P_{off}(y) \log \frac{P_{on}(y)}{P_{off}(y)}$. Therefore we want to select H_L so that $-D(P_{off}||P_{on}) < H_L < D(P_{on}||P_{off})$.

Our complexity results will be obtained using Sanov's theorem [4] to estimate the probability that the algorithms wastes time searching distractor paths. In order to use Sanov's theorem, it is convenient to think of the heuristic as the expected reward of data distributed according to the mixture of P_{on}, P_{off} given by $P_\lambda(y) = P_{on}^{1-\lambda}(y) P_{off}^{\lambda}(y)/Z[\lambda]$ (where $Z[\lambda]$ is a normalization constant). In this section we will consider the special case where $\lambda = 1/2$. This gives the *Bhattacharyya heuristic* $H_L^* = \sum_y P_{\lambda=1/2}(y) \log \frac{P_{on}(y)}{P_{off}(y)}$. (We give it this name because the distribution $P_{\lambda=1/2}$ is associated with the Bhattacharyya bound in statistics [12]). By setting $\lambda = 1/2$ we are essentially choosing a heuristic midway between the target and distractors (generated by P_{on} and P_{off} respectively). (In [2] we will extend the results to deal with other values of λ.)

The Bhattacharyya heuristic is special in two ways. Firstly, it simplifies the analysis. Secondly, and more importantly, we can prove stronger results about convergence if the Bhattacharyya heuristic is used (although this may reflect limitations in our proofs). As we show in [2], if the algorithm converges using one of the alternative heuristics then it will also converges with the Bhattacharyya heuristic, but the reverse is not necessarily true.

The Bhattacharyya heuristics H_L^*, H_P^* are the expected rewards per segment:

$$H_L^* = \vec{\phi}_{Bh} \cdot \vec{\alpha}, \qquad H_P^* = \vec{\psi}_{Bh} \cdot \vec{\beta}, \tag{3.1}$$

with respect to the distributions ϕ_{Bh}, ψ_{Bh}:

$$\begin{aligned} \phi_{Bh}(y) &= \frac{\{P_{on}(y)\}^{1/2}\{P_{off}(y)\}^{1/2}}{Z_\phi}, \\ \psi_{Bh}(t) &= \frac{\{P_{\Delta G}(t)\}^{1/2}\{U(t)\}^{1/2}}{Z_\psi}, \end{aligned} \tag{3.2}$$

where Z_ϕ, Z_ψ are normalization constants.

We first put an upper bound on the probability that any completely false segment is searched.

Let $A_{n,i}$ be the set of subpaths of length n that belong to F_i. Then we have the following result:

THEOREM 1. *The probability that A* searches the last segment of a particular subpath in $A_{n,i}$ is less than or equal to $Pr\{\exists\ m : S_{off}(n) \geq S_{on}(m)\}$.*

Proof. By definition of A*, a **necessary** condition for the segment to be searched is that its A* reward (including the heuristic) is better than the A* reward of at least one segment on the target path. This is because the A* algorithm always maintains a queue of nodes to explore and searches the node segment with highest reward. The algorithm is initialized at the start of the target path and so an element of the target path will always lie in the queue of nodes that A* considers searching. (This condition is not sufficient to ensure that the segment is searched – so we are only obtaining an upper bound). □

We now bound $Pr\{\exists\ m : S_{off}(n) \geq S_{on}(m)\}$ by something we can evaluate.

THEOREM 2.

$$Pr\{\exists\ m : S_{off}(n) \geq S_{on}(m)\} \leq \sum_{m=0}^{\infty} Pr\{S_{off}(n) \geq S_{on}(m)\}.$$

Proof. Boole's inequality. □

We now proceed to find a bound on $Pr\{S_{off}(n) \geq S_{on}(m)\}$. This is done using Sanov's theorem (see [2] for details). It will show that this probability falls-off exponentially with n, m (provided certain parameters are positive).

THEOREM 3.

$$Pr\{S_{off}(n) \geq S_{on}(m)\} \leq \{(n+1)(m+1)\}^{J^2Q^2} 2^{-(n\Psi_1 + m\Psi_2)},$$

where $\Psi_1 = D(\vec{\phi}_{Bh}||P_{off}) + D(\vec{\psi}_{Bh}||U)$ *and* $\Psi_2 = D(\vec{\phi}_{Bh}||P_{on}) + D(\vec{\psi}_{Bh}||P_{\Delta G})$.

Proof. The proof is an application of Sanov's theorem, see [2], applied to the product space of types of $P_{on}, P_{off}, P_{\Delta G}, U$. Define:

$$\begin{aligned} E = \{(\vec{\phi}^{off}, \vec{\psi}^{off}, \vec{\phi}^{on}, \vec{\psi}^{on}) : n\{\vec{\phi}^{off} \cdot \vec{\alpha} - H_L^* + \vec{\psi}^{off} \cdot \vec{\beta} - H_p^*\} \\ \geq m\{\vec{\phi}^{on} \cdot \vec{\alpha} - H_L^* + \vec{\psi}^{on} \cdot \vec{\beta} - H_p^*\}\}. \end{aligned} \tag{3.3}$$

(i.e. E is the set of all histograms corresponding to partial off paths with higher A* reward than the partial on path).

Sanov's theorem gives a bound in terms of the $\phi^{\text{off}}, \psi^{off}, \phi^{on}, \psi^{on}$ that minimize:

$$
\begin{aligned}
f(\vec{\phi}^{off}, \vec{\psi}^{off}, \vec{\phi}^{on}, \vec{\psi}^{on}) &= nD(\vec{\phi}^{off}||P_{off}) + nD(\vec{\psi}^{off}||U) \\
&+ mD(\vec{\phi}^{on}||P_{on}) + mD(\vec{\psi}^{on}||P_{\Delta G}) \\
&+ \tau_1\left\{\sum_y \phi^{off}(y) - 1\right\} + \tau_2\left\{\sum_t \psi^{off}(t) - 1\right\} \\
&+ \tau_3\left\{\sum_y \phi^{on}(y) - 1\right\} + \tau_4\left\{\sum_t \psi^{on}(t) - 1\right\} \\
&+ \gamma\{m\{\vec{\phi}^{on}\cdot\vec{\alpha} - H_L^* + \vec{\psi}^{on}\cdot\vec{\beta} - H_p^*\} \\
&- n\{\vec{\phi}^{off}\cdot\vec{\alpha} - H_L^* + \vec{\psi}^{off}\cdot\vec{\beta} - H_p^*\}\},
\end{aligned}
\tag{3.4}
$$

where the τ's and γ are Lagrange multipliers. This function $f(.,.,.,.)$ is known to be convex so there is a unique minimum. Observe that $f(....)$ consists of four terms of form $nD(\vec{\phi}^{off}||P_{off}) + \tau_1\{\sum_y \phi^{off}(y) - 1\} - n\gamma\vec{\phi}^{off}\cdot\vec{\alpha}$ which are coupled only by shared constants. These terms can be minimized separately to give:

$$
\begin{aligned}
\vec{\phi}^{off*} &= \frac{P_{on}^{\gamma} P_{off}^{1-\gamma}}{Z[1-\gamma]}, \qquad & \vec{\phi}^{on*} &= \frac{P_{on}^{1-\gamma} P_{off}^{\gamma}}{Z[\gamma]}, \\
\vec{\psi}^{off*} &= \frac{P_{\Delta G}^{\gamma} U^{1-\gamma}}{Z_2[1-\gamma]}, \qquad & \vec{\psi}^{on*} &= \frac{P_{\Delta G}^{1-\gamma} U^{\gamma}}{Z_2[\gamma]},
\end{aligned}
\tag{3.5}
$$

subject to the constraint given by equation (3.3).

By inspection, the unique solution occurs when $\gamma = 1/2$. In this case:

$$
\vec{\phi}^{off*}\cdot\vec{\alpha} = H_L^* = \vec{\phi}^{on*}\cdot\vec{\alpha}, \qquad \vec{\psi}^{off*}\cdot\vec{\beta} = H_P^* = \vec{\psi}^{on*}\cdot\vec{\beta}. \tag{3.6}
$$

The solution occurs at $\vec{\phi}^{on*} = \vec{\phi}^{off*} = \vec{\phi}_{Bh}$ and at $\vec{\psi}^{on*} = \vec{\psi}^{off*} = \vec{\psi}_{Bh}$. Substituting into the Sanov bound gives the result. □

From Theorem 3, it is a direct summation, and application of Theorem 2, to obtain:

THEOREM 4. *$Pr\{\exists\ m : S_{off}(n) \geq S_{on}(m)\} \leq \sum_{m=0}^{\infty} Pr\{S_{off}(n) \geq S_{on}(m)\} \leq (n+1)^{J^2Q^2} C_2(\Psi_2) 2^{-n\Psi_1}$, where Ψ_1, Ψ_2 are specified in Theorem 3, and*

$$
C_2(\Psi_2) = \sum_{m=0}^{\infty} (m+1)^{J^2Q^2} 2^{-m\Psi_2}. \tag{3.7}
$$

Theorem 4 shows that the probability of exploring a particular distractor path to depth n falls off exponentially with n.

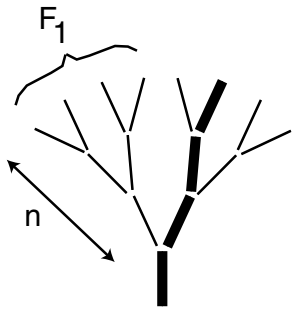

FIG. 9. *This figure illustrates Theorem 5. F_1 has at most Q^n segments at depth n.*

We now compute the expected number of false segments that are searched at depth n in the tree. There is a factor of $Q^n - 1$ segments at this depth which we can bound above by Q^n.

THEOREM 5. *Provided $\Psi_1 > \log Q$, the expected number of segments searched in F_1 is less than or equal to $C_2(\Psi_2)C_1(\Psi_1 - \log Q)$, where:*

$$C_1(\Psi_1 - \log Q) = \sum_{n=1}^{\infty} n(n+1)^{J^2Q^2} 2^{-n(\Psi_1 - \log Q)}. \tag{3.8}$$

Proof. There are at most Q^n segments at depth n in F_1. Using Theorem 4, the expected number of segments explored is less than or equal to $\sum_{n=0}^{\infty} Q^n(n+1)^{J^2Q^2} C_2(\Psi_2)2^{-n\Psi_1}$. Sum the series. It will not converge unless $\Psi_1 > \log Q$. □

Finally, by the recursive structure of the tree we see that the number of segments explored in the sets $F_2, F_3, ...$ must be less than, or equal to, the number explored in F_1. (We simply eliminate the first few segments, which are in common with the target path, and perform the same argument as above). This yields our main result:

THEOREM 6. *The expected number of segments explored by an A* algorithm using the Bhattacharyya heuristic is bounded above by $NC_2(\Psi_2)C_1(\Psi_1 - \log Q)$, provided $\Psi_1 > \log Q$.*

The algorithm is expected to explore $O(N)$ segments and the coefficients $C_2(\Psi_2)C_1(\Psi_1 - \log Q)$ rapidly decrease as Ψ_2 and $\Psi_1 - \log Q$ increase.

In addition, we can estimate the expected error of how much our final estimate differs from the target path. Note that *this is not the same as the error with respect to the MAP estimate.* The error is defined as the number of incorrect segments on the path estimated by A*, see Figure 10.

THEOREM 7. *The expected error is bounded above by $C_2(\Psi_2)C_1(\Psi_1 - \log Q)$, provided $\Psi_1 > \log Q$.*

Proof. We measure the error in terms of the expected number of off-road segments. The expected error can be bounded above by $\sum_{n=0}^{\infty} Pr(n)n$, where $Pr(n)$ is the probability that A* will explore a path in F_{N+1-n} to

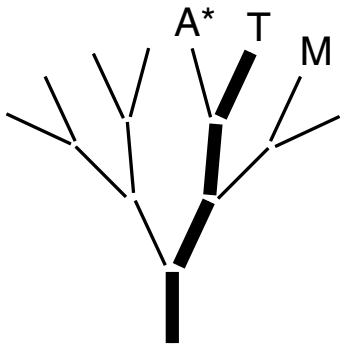

FIG. 10. *Illustrates Theorem 7. A* marks the path found by the A* algorithm. M marks the MAP estimate of the target path position. T marks the target path. Observe that A* has an error of one segment and the MAP has an error of two.*

the end. There are Q^n such paths. For each path, the probability that we explore it to the end is bounded above by $\sum_{m=0}^{\infty} Pr\{S_{off}(n) \geq S_{on}(m)\} \leq (n+1)^{J^2Q^2} C_2(\Psi_2) 2^{-n\Psi_1}$, using Theorem 4. Therefore the expected error is bounded above by $\sum_{n=0}^{\infty} nQ^n(n+1)^{J^2Q^2} C_2(\Psi_2) 2^{-n\Psi_1}$, which was summed in Theorem 5. Hence result. □

The condition $\Psi_1 > \log Q$ is related to the order parameter K_B given by equation (2.9). It is straightforward algebra to check that $K_B = \Psi_1 + \Psi_2 - \log Q$. Therefore the condition $\Psi_1 > \log Q$ implies that $K_B > 0$ ($\Psi_2 > 0$ by definition) which ensures that the expected number of paths in F_1 with rewards higher than the target path will fall to zero as $N \mapsto \infty$.

4. Sorting the queue in linear expected time. We have shown that the expected number of nodes searched is linear in N. But the convergence rate of the algorithm will also depend on how much time is required to sort the queue of nodes that we want to expand. In this section, we prove that the expected time to sort the queue nodes is constant.

We use a simple linked list data structure where we order the queue nodes according to their rewards (instead of a more sophisticated data structure, like a heap – see, for example, [5, 3]). A* proceeds by expanding the top node (the one with highest A* reward) and must then adjust the queue to accommodate its children. We now show that the expected sort time, which is required to place the children in their correct positions in the queue, is a constant. To do this, we note that the children nodes have A^* rewards that are smaller than the top node by at most Λ, where $\Lambda = H_L + H_P - \min_y \log P_{on}(y)/P_{off}(y) - \min_t \log P_{\Delta G}(t)/U(t)$ (note that $\Lambda > 0$). We therefore only have to compare the rewards of the children with nodes whose rewards are within Λ of the top node. As we will show, the expected number of these nodes is constant. This gives the following theorem.

THEOREM 8. *The expected sorting rate is constant (i.e. independent of the size N of the problem).*

Proof. The expected sorting rate is equal to Q times the expected number of nodes in the sort queue which have rewards within Λ of the top node. The reward of the top node is guaranteed to be greater than, or equal to, the reward r_T of the longest target subpath in the queue. Let this longest target subpath have length n. To prove that the expected sorting time is constant it suffices to show that the expected number of paths in the queue with rewards greater than $r_T - \Lambda$ is constant. This requires computing the probabilities that subpaths in $F_1, ..., F_n$ have rewards higher than $r_T - \Lambda$ and bounding the expected number of such subpaths. (We do not need to consider paths in $F_i, \quad i > n$ because, by definition of n, they involve children of nodes in the queue and so cannot be in the queue.) We can bound these probabilities using Sanov's theorem and then bound the expected number of nodes by summing exponential series. The details are given in [2]. □

5. Conclusion. The goal of this paper is to point out that Bayesian formulation of inference problems leads to a probability distribution on the ensemble of problem instances. Analysis of this ensemble can give complexity results and, in other work [20, 21], algorithm-independent results.

As a specific example, we analyzed the Geman and Jedynak [6] theory for road tracking. We were able to demonstrate linear (in the road size) expected convergence for a class of ensembles even though the worst case performance is exponential. This agrees with previous work [17] where we analyzed a block-pruning search strategy motivated by [11].

Acknowledgments. We want to acknowledge funding from NSF with award number IRI-9700446, from the Center for Imaging Sciences funded by ARO DAAH049510494, and from the Smith-Kettlewell core grant, and the AFOSR grant F49620-98-1-0197 to A.L.Y. Lei Xu drew our attention to Pearl's book on heuristics and we thank Abracadabra books for obtaining a second hand copy for us. We would also like to thank Dan Snow and Scott Konishi for helpful discussions as the work was progressing and Davi Geiger for providing useful stimulation. David Forsyth, Jitendra Malik, Preeti Verghese, Dan Kersten, Suzanne McKee and Song Chun Zhu gave very useful feedback and encouragement. Finally, we wish to thank Tom Ngo for drawing our attention to the work of Cheeseman and Selman.

REFERENCES

[1] P. CHEESEMAN, B. KANEFSKY, AND W. TAYLOR. "Where the Really Hard Problems are." In *Proc. 12th International Joint Conference on A.I.*. **1**: 331–337. Morgan-Kaufmann. 1991.

[2] J.M. COUGHLAN AND A.L. YUILLE. "Bayesian A* Tree Search with Expected O(N) Node Expansions for Road Tracking." Neural Computation. In press. 2002.

[3] J.M. COUGHLAN, D. SNOW, C. ENGLISH, AND A.L. YUILLE. "Efficient Deformable Template Detection and Localization without User Initialization." *Computer Vision and Image Understanding.* **78**: 303–319. June 2000.

[4] T.M. COVER AND J.A. THOMAS. **Elements of Information Theory**. Wiley Interscience Press. New York. 1991.

[5] D. GEIGER AND T-L LIU. "Top-Down Recognition and Bottom-Up Integration for Recognizing Articulated Objects." In *Proceedings of the International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition.* Eds. M. Pellilo and E. Hancock. Venice, Italy. Springer-Verlag. May. 1997.

[6] D. GEMAN. AND B. JEDYNAK. "An active testing model for tracking roads in satellite images." *IEEE Trans. Patt. Anal. and Machine Intel.* **18**(1): 1–14. January 1996.

[7] G.R. GRIMMETT AND D.R. STIRZAKER. **Probability and Random Processes**. Clarendon Press. Oxford. 1992.

[8] R.M. KARP AND J. PEARL. "Searching for an Optimal Path in a Tree with Random Costs." *Artificial Intelligence.* **21**(1,2): 99–116. 1983.

[9] D.C. KNILL AND W. RICHARDS (Eds). **Perception as Bayesian Inference**. Cambridge University Press. 1996.

[10] S. KONISHI, A.L. YUILLE, J.M. COUGHLAN, AND S.C. ZHU. "Fundamental Bounds on Edge Detection: An Information Theoretic Evaluation of Different Edge Cues." *Proc. Int'l conf. on Computer Vision and Pattern Recognition*, 1999.

[11] J. PEARL. **Heuristics**. Addison-Wesley. 1984.

[12] B.D. RIPLEY. **Pattern Recognition and Neural Networks**. Cambridge University Press. 1996.

[13] S. RUSSELL AND P. NORVIG. "Artificial Intelligence: A Modern Approach. Prentice-Hall. 1995.

[14] B. SELMAN AND S. KIRKPATRICK. "Critical Behavior in the Computational Cost of Satisfiability Testing." Artificial Intelligence. **81**(1–2): 273–295. 1996.

[15] V.N. VAPNIK. **Statistical Learning Theory**. John Wiley and sons. New York. 1998.

[16] P.H. WINSTON. **Artificial Intelligence**. Addison-Wesley Publishing Company. Reading, Massachusetts. 1984.

[17] A.L. YUILLE AND J.M. COUGHLAN. "Convergence Rates of Algorithms for Visual Search: Detecting Visual Contours." In *Proceedings NIPS'98.* 1998.

[18] A.L. YUILLE AND J. COUGHLAN. "An A* perspective on deterministic optimization for deformable templates." *Pattern Recognition.* **33**(4): 603–616. April 2000.

[19] A.L. YUILLE AND J.M. COUGHLAN. "Convergence Rates of Algorithms for Visual Search: Detecting Visual Contours." In **Advances in Neural Information Processing Systems 11**. Eds. M.S. Kearns, S.A. Solla, and D.A. Cohn. pp. 641–647. 1999.

[20] A.L. YUILLE AND J.M. COUGHLAN. "Fundamental Limits of Bayesian Inference: Order Parameters and Phase Transitions for Road Tracking." *Transactions on Pattern Analysis and Machine Intelligence.* PAMI. **22**: 1–14. February 2000.

[21] A.L. YUILLE, J.M. COUGHLAN, Y-N. WU, AND S.C. ZHU. "Order Parameters for Minimax Entropy Distributions: When does high level knowledge help?" *International Journal of Computer Vision.* **41**(1/2): 9–33. 2001.