Efficient methods for object recognition using the constellation model

R. Fergus[†] M. Weber[‡] P. Perona[†‡§]

[†]Dept. of Electrical Engineering
[‡]Dept. of Computation and Neural Systems
California Institute of Technology
Pasadena, CA 91125, U.S.A.
{fergus,mweber,perona}@vision.caltech.edu

[§]Università di Padova
Italy

Abstract

We present efficient methods for object recognition, using the constellation model. This model represents objects as constellations of rigid features, with the variability between them being represented by a joint probability density function. Using simple, exhaustive methods, the computational requirements for this approach quickly become prohibitive. By using A search methods, accompanied by some heuristics, large performance improvements are achievable in recognition, making real-time recognition on large images possible, using complex object models.*

1. Introduction

One of the most important functions of vision is recognizing the objects that surround us. A number of different approaches have been proposed for recognizing individual objects and/or object categories [1][2][3][4]. The main three issues that must be solved in a recognition system are representation, learning and detection. The challenge of representation is to develop models that can accommodate intrinsic object variability within a class, and at the same time be invariant with respect to rigid transformations (translation, rotation), scaling and lighting conditions. The challenge of learning is one of estimating representation parameters from possibly cluttered and fragmentary training information. The challenge of detection and recognition is matching models of objects, and object categories, to images in the presence of occlusion and spurious clutter. While learning may be slow because it typically happens off-line, detection and recognition must be fast, if possible real-time.

We examine here the problem of efficient detection of objects in images containing large amounts of clutter and possibly occlusion. We explore the properties of the "constellation" model proposed by Burl et al.[5][6]. This model presents a number of appealing properties in that it treats deformations, occlusion and the effects of image clutter in a principled probabilistic framework. Its main shortcoming is that detection is intrinsically combinatorial: all possible arrangements of the detected features have to be explored. Burl et al. have proposed an efficient method for searching only the most plausible such constellations. However, their method is based on a heuristic whose validity is difficult to control. Additionally, it is not clear that Burl et al's method is as efficient as allowed by the structure of the problem.

Our approach consists in applying a classical search technique of artificial intelligence, A*., to the problem of constellation searching.

In section 2 we review the constellation model. Then in section 3 we derive some efficient approaches to recognition. In section 4 and 5 we discuss the implementation of our methods and then demonstrate their results. Finally, in section 6, we present our conclusions.

2. The Constellation Model

We now cover the main aspects of the constellation model. For more details, see [7][8]. Objects are modeled as collections of rigid parts, each of which is detected by a corresponding detector during recognition. The part detection stage therefore transforms an entire image into a collection of parts. Some of those parts might correspond to an instance of the target object class (the *foreground*), while others stem from background clutter or are simply false detections (the *background*). Throughout this paper, the only information associated with an object part is its position in the image and its identity or part *type*. We assume that there are T different types of parts. The positions of all parts extracted from one image can be summarized in a matrix-like form,

$$\mathcal{X} = \left(\begin{array}{c} x_{11} x_{12}, \ldots, x_{1N_1} \\ x_{21} x_{22}, \ldots, x_{2N_2} \\ \vdots \\ x_{T1} x_{T2}, \ldots, x_{TN_T} \end{array} \right. ,$$

The t^{th} row contains the locations of detections of part type t, where every entry, x_{ij}, is a two-dimensional vector. If we now assume that an object is composed of F different parts,[1] we need to be able to indicate which parts in $\mathcal{X}$ correspond to the foreground (the object of interest). For this we use the vector $\mathbf{h}$, a set of indices, with $h_i = j,\ j > 0$, indicating that point x_{ij} is a foreground point. If an object part is not contained in $\mathcal{X}$, because it is occluded or otherwise undetected, the corresponding entry in $\mathbf{h}$ will be zero. We call $\mathbf{h}$ a *hypothesis*, since we will use it to hypothesize that certain parts of $\mathcal{X}$ belong to the foreground object.

We can now define a generative probabilistic model through the joint probability density $p(\mathcal{X}, \mathbf{h})$.

2.1. Model Details

In order to provide a detailed parameterization of the model, we introduce two auxiliary variables, $\mathbf{d}$ and $\mathbf{n}$. The binary vector $\mathbf{d}$ encodes information about which parts have been detected and which have been missed or occluded. Hence, $d_f = 1$ if $h_f > 0$ and $d_f = 0$ otherwise. The variable $\mathbf{n}$ is also a vector, where n_t shall denote the number of *background* candidates included in the t^{th} row of $\mathcal{X}$. Since both variables are completely determined by $\mathbf{h}$ and the size of $\mathcal{X}$, we have $p(\mathcal{X}, \mathbf{h}) = p(\mathcal{X}, \mathbf{h}, \mathbf{n}, \mathbf{d})$. Since we assume independence between foreground and background, and, thus, between $p(\mathbf{n})$ and $p(\mathbf{d})$, we decompose in the following way

$$p(\mathcal{X}, \mathbf{h}, \mathbf{n}, \mathbf{d}) = p(\mathcal{X}|\mathbf{h}, \mathbf{n})\, p(\mathbf{h}|\mathbf{n}, \mathbf{d})\, p(\mathbf{n})\, p(\mathbf{d}).$$

The probability density over the number of background detections can be modeled by a Poisson distribution,

$$p(\mathbf{n}) = \prod_{t=1}^{T} \frac{1}{n_t!} (M_t)^{n_t} e^{-M_t},$$

where M_t is the average number of background detections of type t per image. This conveys the assumption of independence between part types in the background and the idea that background detections can arise at any location in the image with equal probability, independently of other locations. Admitting a different M_f for every part type allows us to model the different detector statistics.

Depending on the number of parts, F, we can model the probability $p(\mathbf{d})$ either as an explicit table (of length 2^F) of joint probabilities, or, if F is large, as F independent probabilities, governing the presence or absence of an individual model part. Since the joint treatment gives a powerful model, we use use this approach.

The density $p(\mathbf{h}|\mathbf{n}, \mathbf{d})$ is modeled by,

$$p(\mathbf{h}|\mathbf{n}, \mathbf{d}) = \begin{cases} \frac{1}{\prod_{f=1}^{F} N_f^{d_f}} & \mathbf{h} \in \mathcal{H}(\mathbf{d}, \mathbf{n}) \\ 0 & \text{other } \mathbf{h} \end{cases}$$

[1]To simplify notation, we only consider the case where $F = T$. The extension to the general case ($F \geq T$) is straightforward.

where $\mathcal{H}(\mathbf{d}, \mathbf{n})$ denotes the set of all hypotheses consistent with $\mathbf{d}$ and $\mathbf{n}$, and N_f denotes the total number of detections of the type of part f. This expresses the fact that all consistent hypotheses, the number of which is $\prod_{f=1}^{F} N_f^{b_f}$, are equally likely in the absence of information on the part locations.

Finally, we use

$$p(\mathcal{X}|\mathbf{h}, \mathbf{n}) = p_{\mathbf{fg}}(\mathbf{x}_{fg})\, p_{\mathbf{bg}}(\mathbf{x}_{bg}),$$

where we defined as the coordinates of all foreground detections (observed and missing) and $\mathbf{x}_{bg}$ as the coordinates of all background detections. Here we have made the important assumption that the foreground detections are independent of the background. In our experiments, $p_{\mathbf{fg}}(\mathbf{x}_{fg})$ is modeled as a joint Gaussian with mean μ and covariance Σ.

Note that, so far, we have modeled only *absolute* part positions in the image, making the model translation-variant. It is possible to make the model translation-invariant (Gaussians remain Gaussians under translation), but the formulation is intricate so will not be presented here. For further discussion, see [9].

The positions of the background detections are modeled by a uniform density,

$$p_{\mathbf{bg}}(\mathbf{x}_{bg}) = \prod_{t=1}^{T} \frac{1}{A^{n_t}},$$

where A is the total image area.

2.2. Classification

Our overall objective is to classify images into either object present ($\mathcal{O}_1$) or object absent ($\mathcal{O}_0$). This decision R is made by comparing the Bayesian posterior probabilities:

$$R = \frac{p(\mathcal{O}_1|\mathcal{X})}{p(\mathcal{O}_0|\mathcal{X})} = \frac{p(\mathcal{X}|\mathcal{O}_1)\ p(\mathcal{O}_1)}{p(\mathcal{X}|\mathcal{O}_0)\ p(\mathcal{O}_0)}$$

If we assume the priors to be equal, the decision rests with the likelihoods. If $R > 1$ an object is detected, if not, then no object is detected. The likelihood can be expanded:

$$\begin{aligned} p(\mathcal{X}|\mathcal{O}) &= \sum_{\mathbf{h}\in\mathcal{H}} p(\mathcal{X}, \mathbf{h}|\mathcal{O}) \\ &= \sum_{\mathbf{h}\in\mathcal{H}} p(\mathcal{X}|\mathbf{h}, \mathbf{n})\ p(\mathbf{h}|\mathbf{n}, \mathbf{d})\ p(\mathbf{n})\ p(\mathbf{d}|\mathcal{O}). \end{aligned}$$

In the cases of $\mathcal{O}_0$, only one hypothesis exists: $\mathbf{h}_0 = \mathbf{0}$. $\mathbf{h}_0$ is denoted the *null hypothesis*. Hence

$$\begin{aligned} R &= \frac{p(\mathcal{X}|\mathcal{O}_1)}{p(\mathcal{X}|\mathcal{O}_0)} = \frac{\sum_{\mathbf{h}\in\mathcal{H}} p(\mathcal{X}, \mathbf{h}|\mathcal{O}_1)}{p(\mathcal{X}, \mathbf{h}|\mathcal{O}_0)} && (1) \\ &= \frac{\sum_{\mathbf{h}\in\mathcal{H}} p(\mathcal{X}|\mathbf{h}, \mathbf{n})\ p(\mathbf{h}|\mathbf{n}, \mathbf{d})\ p(\mathbf{n})\ p(\mathbf{d}|\mathcal{O}_1)}{p(\mathcal{X}|\mathbf{h}_0, \mathbf{n})\ p(\mathbf{h}_0|\mathbf{n}, \mathbf{d})\ p(\mathbf{n})\ p(\mathbf{d}|\mathcal{O}_0)} && (2) \end{aligned}$$

2.3. Learning

Before we can classify any images however, we must train an object model on a set of training images. We must first decide on which parts we will use in our model. Then we must learn the parameters of the various probability densities, introduced above. The first problem is tackled using a "greedy" approach of trying different parts on a validation set and changing the parts, always seeking a better performance on the validation set. The second problem is tackled using *expectation maximization* (EM) [10].

2.4. Computational Issues

From a computational point of view, the constellation model does not scale well with either the number of features, F, or the average number of detections, a. The number of hypotheses which must be summed over is $O(a^F)$. Since the evaluation of each hypothesis takes unit time, the calculation of the $p(\mathcal{X}|\mathcal{O})$ is also $O(a^F)$.

The learning process is the most time consuming since we need to try many different combinations of parts to find the best combination. Each set of parts needs 100 EM iterations to converge. With current hardware, we can evaluate $O(10^5)$ hypothesis/second. In practice this means we are restricted to a and F in the region: $a = 5$, $F = 5$. Both of these values represent serious limitations.

Considering the number of features, complex objects may require many features to learn a representative model. The recognition performance on simple objects would be improved by using more features.

The limitation on a is probably more serious however, since $a = 5$ requires very reliable feature detectors to ensure there are not too many false alarms. Failure of the feature detectors is the major source of misclassification error, so by being able to cope with $a > 5$ the recognition performance will be improved. In addition, with multi-mega-pixel cameras now widely available, the total size of images that we may wish to recognize is getting very large and to constrain ourselves to only 5 detections over a 2000x1500 image is unrealistic.

Increasing the maximum practical values of a and F is achieved by reducing the number of hypotheses we actually consider, therefore assuming that a portion of them are irrelevant and therefore need not be considered.

Since learning the model is a hard problem, we will only consider detection. In detection, we only have to sum over the hypotheses once in order to make a decision as to the presence/absence of an object. In the process of doing this, we can also localize it within the image.

Although we have a large number of hypotheses ($O(10^{6-9}$), the dynamic range their likelihoods is very large ($O(10^{100})$). So there many orders of magnitude between the best and worst hypotheses. A feel for their distribution can be obtained by looking at Fig.4. Since we will be summing over them, it is clear that only the very few best hypotheses will contribute significantly to the decision R.

While it is true that many weak hypotheses could combine to make $R > 1$, it is our experience that when the object is present, R far exceeds the threshold, typically by many orders of magnitude. Therefore, the decision R can be safely be made by only computing the best few hypotheses.

3. Efficient search methods

To make the recognition process quicker, we need to efficiently search through all the hypotheses to find the best ones.

The simplest way to find these few hypotheses is to exhaustively search through all of them and retain the best. It is guaranteed to find the best hypothesis, but saves us nothing, since we still must evaluate all hypotheses. However, this method does give us a benchmark against which we can compare more efficient methods.

We will now consider two efficient methods, both of which rely on visiting the hypotheses in a particular order, only calculating terms that change between them. Expanding the expression for the likelihood $p(\mathcal{X}, \mathbf{h}|\mathcal{O}_1)$, we see that for each hypothesis, we have to calculate:

$$\begin{aligned} p(\mathcal{X}, \mathbf{h}|\mathcal{O}_1) &= p_{\mathrm{fg}}(\mathcal{X}(\mathbf{h})) \prod_{f=1}^{F} p_{\mathrm{Poiss}}(n_f) \prod_{f=1}^{F} A^{-n_f} \\ & \prod_{f=1}^{F} N_f^{-d_f}\, p(\mathbf{d}|\mathcal{O}_1). \end{aligned}$$

For the $p(\mathcal{X}, \mathbf{h}|\mathcal{O}_0)$ the we only need to calculate (the other terms cancel)

$$p(\mathcal{X}, \mathbf{h}_0|\mathcal{O}_0) = \prod_{f=1}^{F} \frac{p_{\mathrm{Poiss}}(N_f)}{A^{N_f}}.$$

By dividing $p(\mathcal{X}, \mathbf{h}|\mathcal{O}_1)$ by $p(\mathcal{X}, \mathbf{h}_0|\mathcal{O}_0)$, we can simplify things considerably.

$$\begin{aligned} p(\mathcal{X}, \mathbf{h}|\mathcal{O}) &\overset{\mathrm{def}}{=} \frac{p(\mathcal{X}, \mathbf{h}|\mathcal{O}_1)}{p(\mathcal{X}, \mathbf{h}_0|\mathcal{O}_0)} \\ &= \sum_{\mathbf{h}\in\mathcal{H}} p_{\mathrm{fg}}(\mathcal{X}(\mathbf{h})) \prod_{f=1}^{F} \left(\frac{A}{M_f}\right)^{d_f} \\ & \prod_{f=1}^{F} \frac{p_{\mathrm{Poiss}}(N_f)}{A^{N_f}} p(\mathbf{d}|\mathcal{O}_1), \end{aligned}$$

using the fact that $\frac{p_{\mathrm{Poiss}}(N)}{p_{\mathrm{Poiss}}(N-1)} = \frac{M}{N}$.

We can also factor p_{fg} by conditioning part positions on the positions of parts corresponding to a smaller index, f.

If a part is missing, we replace the corresponding term with one and omit further conditioning on this part. We obtain

$$\begin{aligned} p(\mathcal{X},\mathbf{h}|\mathcal{O}) &= \prod_{f=1}^{F}\left(p_{\mathrm{fg}}(x_{h_f}|x_{h_1}\dots x_{h_{f-1}})\frac{A}{M_f}\right)^{d_f} \\ &\quad \prod_{f=1}^{F}\frac{p_{\mathrm{Poiss}}(N_f)}{A^{N_f}}p(\mathbf{d}|\mathcal{O}_1). \end{aligned}$$

If $p(\mathbf{d}|\mathcal{O})$ is assumed independent, it can be factored as well and we obtain

$$\begin{aligned} p(\mathbf{d}|\mathcal{O}_1) &= \prod_{f=1}^{F} p(d_f=1)^{d_f}p(d_f=0)^{1-d_f} \\ &= \prod_{f=1}^{F}\left(\frac{p(d_f=1)}{p(d_f=0)}\right)^{d_f}p(d_f=0) \end{aligned}$$

This gives us

$$\begin{aligned} p(\mathcal{X},\mathbf{h}|\mathcal{O}) &= \prod_{f=1}^{F}\left(p_{\mathrm{fg}}(x_{h_f}|x_{h_1}\dots x_{h_{f-1}})\frac{Ap(d_f=1)}{M_f p(d_f=0)}\right)^{d_f} \\ &\quad \prod_{f=1}^{F}\frac{p_{\mathrm{Poiss}}(N_f)}{A^{N_f}}p(d_f=0) \\ &= \prod_{f=1}^{F}\left(p_{\mathrm{fg}}(x_{h_f}|x_{h_1}\dots x_{h_{f-1}})\alpha_f\right)^{d_f}K. \end{aligned}$$

Where $K \stackrel{\mathrm{def}}{=} \prod_{f=1}^{F}\frac{p_{\mathrm{Poiss}}(N_f)}{A^{N_f}}p(d_f=0)$ and $\alpha_f \stackrel{\mathrm{def}}{=} \frac{A}{M_f}\frac{p(d_f=1)}{p(d_f=0)}$ are constant across hypotheses of one image.

Since we assume a joint Gaussian for p_{fg}, we have a simple dependence of the parameters in the factorization:

$$p_{\mathrm{fg}}(x_f|x_1\dots x_{f-1}) = G(x_f|\mu_f(x_1\dots x_{f-1}),\Sigma_f) \stackrel{\mathrm{def}}{=} G_f(x_f).$$

Note that due to the nature of the Gaussian density, only the mean of $p_{\mathrm{fg}}(x_f)$ will depend on the candidate points with lower index, not the covariance matrix.

Assuming that the first ϕ entries of a hypothesis $\mathbf{h}$ have assigned candidate positions or are missing, we can compute an upper bound on $p(\mathcal{X},\mathbf{h}|\mathcal{O})$ that holds independently of which parts, if any, are chosen for $h_{\phi+1}\dots h_F$.

$$\begin{aligned} p(\mathcal{X},\mathbf{h}|\mathcal{O}) &= \prod_{f=1}^{\phi}(G_f\alpha_f)^{d_f}K\cdot\prod_{f=\phi+1}^{F}(G_f\alpha_f)^{d_f} \\ &\le \prod_{f=1}^{\phi}(G_f\alpha_f)^{d_f}K\cdot\prod_{f=\phi+1}^{F}\max_{x_f}(G_f(x_f)\alpha_f,1). \end{aligned}$$

If we note that $\max_{x_f} G_f$ is simply $\frac{1}{(2\pi)^F\sqrt{|\Sigma_f|}} \stackrel{\mathrm{def}}{=} D_f$, we obtain

$$p(\mathcal{X},\mathbf{h}|\mathcal{O}) \le \prod_{f=1}^{\phi}(G_f\alpha_f)^{d_f}K\cdot\prod_{f=\phi+1}^{F}\max(D_f\alpha_f,1). \tag{3}$$

Since D_f does not depend on $(x_1\dots x_{f-1})$, the last factor also needs to be computed only once per image. Note that we assumed $p(\mathbf{d}|\mathcal{O})$ to be independent, but in fact we normally do not make this assumption, so $p(\mathbf{d}|\mathcal{O})$ is a table and cannot be separated into separate terms. However, the same separation into $f\le\phi$ and $f>\phi$ is possible and follows the same principles as above, but is notationally challenging, so is not attempted here.

3.1. Threshold-based searching

We assume that we generate the hypothesis in a particular order, using $\mathbf{h}$ as a counter in such a way that the highest index, h_F, is increased at every step. When it reaches the maximum number of candidates, N_f, for the corresponding part, a “carry over” occurs and h_F will be reset to zero. At the same time the next lower index, h_{F-1} will be increased by one, and so forth. It follows that the first few indices of $\mathbf{h}$ are only updated very rarely.

We can now introduce a threshold which indicates the minimum value that $p(\mathbf{h},\mathcal{X}|\mathcal{O})$ must take on, for the corresponding hypothesis to be included in the computation at hand. By examining a certain “prefix” – a choice for the first ϕ candidates – we can decide if all hypotheses starting with this prefix are to be rejected. The variable ϕ can be dynamical, in order to reject or accept hypotheses with prefixes of any length.

The rejection of negligible hypotheses can be implemented through a look-up table, Ξ, which stores all entities which are constant throughout one signal. Thus,

$$\Xi(\mathbf{d},\phi) = K\cdot\prod_{f=1}^{\phi}\alpha_f^{d_f}\max_{b_{\phi+1,\dots,F}}\prod_{f=\phi+1}^{F}(\alpha_f D_f)^{d_f}.$$

Here, the maximum is taken over all b with the first ϕ bits fixed to the first ϕ bits of the table index.

If we now choose a threshold, T, indicating the minimum value of $p(\mathbf{h},\mathcal{X}|\mathcal{O})$ for a hypothesis to be worthy of consideration, we can perform the necessary check as follows

$$\prod_{f=1}^{\phi}G_f^{d_f} \ge \frac{T}{\Xi(\mathbf{d},\phi)}.$$

If any partial hypothesis with the first ϕ parts candidates chosen does not fulfill the above inequality, all hypotheses with the same “prefix” can be rejected. This enables us to discard a large portion of the hypotheses, so we can quickly find the best ones. For more details of this approach, see [11].

3.2. The Search Tree

A more principled approach to the problem of efficiently finding the best hypotheses exists. The simplification in the previous section allows us to employ an A* search[12], using a search tree of hypotheses. The A* algorithm uses a branch and bound approach, coupled with a upper-bound estimate to the goal to ensure that the optimum is found. Using (3) we obtain our upper-bound estimate of $p(\mathcal{X}, \mathbf{h}|\mathcal{O})$ which forms the admissible heuristic for the A* search. $p(\mathcal{X}, \mathbf{h}|\mathcal{O})$ is also cleanly partitioned neatly into $f \leq \phi$ and $f > \phi$ therefore enabling us to consider the each feature independently.

The search tree, as illustrated in Fig.1, works as follows: Let the root node be a hypotheses where none of the features have yet been selected (denoted by an "X" in Fig.1). At the bottom of the tree, the leaves are complete hypotheses where all features have been determined (i.e. are no "X"'s in the hypothesis any longer). Zeros in a given hypothesis indicate the corresponding feature is missing. The intermediate levels are partially completed hypotheses, where at level ϕ we set $h_1 \ldots h_\phi$ When opening a node at level ϕ, we have $N_{\phi+1} + 1$ possible selections for the next feature (the extra one is for the feature being missing), so the branching factor of the tree is given by N_f. In the figure, we see that $N_1 = 3, N_2 = 2, N_3 = 3$. This corresponds to the number of nodes that are opened up at each level.

In order to decide how we should traverse the tree, each node has an overall score, T, which is equal to $A + B$ defined below, (see (3)).

$$
\begin{aligned}
A &\overset{\text{def}}{=} \log \prod_{f=1}^{\phi} (G_f \alpha_f)^{d_f} = \sum_{f=1}^{\phi} \log (G_f \alpha_f)^{d_f} \\
B &\overset{\text{def}}{=} \log K \prod_{f=\phi+1}^{F} \max(D_f \alpha_f, 1) \\
&= \log K + \sum_{f=\phi+1}^{F} \log \max(D_f \alpha_f, 1)
\end{aligned}
$$

Intuitively, A is the score the node has accumulated in moving from the root to it's current point. B is the best possible score that a leaf (a hypothesis) belonging to the subtree derived the current node could get.

So the algorithm to find the best H hypotheses is:

```
1. Form an empty array, $P$, of length $H$.
2. Form a queue, $Q$, consisting of the root hypothesis (i.e. all features undefined).
3. Until $P$ is full:
    (a) Remove the node, $N^*$ at the head of $Q$.
    (b) If $N^*$ is a leaf node (valid hypothesis), insert it into $P$, in the next blank slot.
    (c) If not:
        i. Expand $N^*$ to give all valid nodes derived from it.
        ii. For each new node:
            1. Calculate $T$.
            2. Insert into $Q$.
    (d) Sort $Q$, in descending order of $T$.
4. Return $P$, which contains the $H$ best hypotheses in order.
```

This algorithm is demonstrated in the Fig.1. By each node, there are toy examples for the values of A,B and T, illustrating its operation.

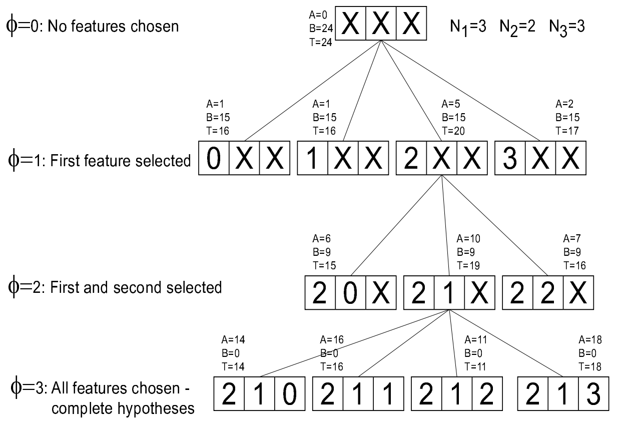


Figure 1: Finding the best hypothesis in the search tree using A*.

3.3. Space Thresholding

In 3(c)ii of the A* search tree algorithm, we blindly put every valid child node into the queue. However, if the branching factor of the tree is very high, the queue will become large, very quickly. In addition to requiring a large amount of memory, the queue will also be slower to access and sort.

However, we can exploit the fact that if we have a missing feature in the search tree, which can act as a lower-bound for the child nodes. If we are at level $\phi - 1$ in the tree and considering it's child nodes at level ϕ, we take our worst case scenario to be if all following features are missing, i.e. $h_f = 0$ for $f = \phi \ldots F$. Let us call this hypothesis $\mathbf{h}_w$. Now if, when considering a particular detection, say, $x_{\phi j}$, gives an upper-bound that is worse than the worst case scenario, then we can be sure that we would rather all subsequent detections be missing than choose detection j. Hence we can prune the sub-tree eminiating from detection j.

The best hypothesis will still be the same as before, but subsequent hypotheses are not guaranteed to be the same, since they might have been pruned. However, if we have reliable detectors, missing features will result in a very low likelihood, so the pruned hypotheses will not be relevant to calculating R.

$$p(\mathcal{X}, \mathbf{h}_w|\mathcal{O}) = \prod_{f=1}^{\phi-1} (G_f \alpha_f)^{d_f} K$$

Now for $x_{\phi j}$ to not be discarded, we require:

$$\prod_{f=1}^{\phi-1} (G_f \alpha_f)^{d_f} \cdot G_\phi(x_{\phi j})\alpha_\phi \cdot K \cdot \prod_{f=\phi+1}^{F} \max(D_f \alpha_f, 1) \leq p(\mathcal{X}, \mathbf{h}_w|\mathcal{O})$$

So

$$G_\phi(x_{\phi j}) \leq \frac{1}{\prod_{f=\phi+1}^{F} \max(D_f \alpha_f, 1)} \tag{4}$$

Once we have calculated the parameters of G_ϕ, μ and Σ, we simply compare G_ϕ to the fixed threshold given by the right-hand side of (4). This can be done efficiently by only computing the exponent of G_ϕ since the determinant term is fixed.

4. Experimental Method

We now have three different techniques for finding the best hypotheses:

1. Exhaustive search

2. Threshold-based search

3. A* search with thresholding

These were evaluated on 100 face images (the same data set as used in [8]) where 5 models were hand-trained on them with 5,6,7,8,9 and 10 features correspondingly. The models were hand-trained, since learning them automatically, as in [8], is not practical for more than 6 features. The average number of detections per image was also controlled, but there was some degree of variance in the number of detections, since some images had more background detections than others. The time per image was calculated by averaging over the 100 images, to take into account these variations. The experiments were run on 750Mhz Pentium III machines running Linux.

Some discussion is required about the 2nd and 3rd methods, since they have parameters that will influence their performance. In implementing them, various optimizations can be made. The vast majority of variables are only dependent on $\mathbf{d}$. Now, even if we choose $\mathbf{d}$ not to be feature-wise independent, it still means there are only 2^F possible values for them. They can be pre-computed beforehand, so we only need to evaluate the Gaussian foreground terms in the main iteration.

4.1. Threshold-based search

The obvious drawback to this method is that the value of the threshold is somewhat arbitrary and must be chosen with care to avoid either including all hypotheses or excluding all. The value of the threshold will depend on the range of the likelihoods, which can span many orders of magnitude and is very much dependent on the number of features and other factors. The setting was done by running on a few sets of the images and choosing a conservative value that would definitely work for all images. The sensitivity of the threshold value is also investigated. In fact, as we change the other parameters (number of features, average number of detections), the relative value of the threshold changes, so making it very difficult to compare its performance fairly.

4.2. A* search

Here the number of hypotheses we want to extract from the tree is important. The more we wish to extract, the more nodes we must open and therefore the higher the cost. We chose the number to extract, H to be equal to F, since there would be around $F + 1$ good hypotheses - hopefully one actual face detection and F with one of the features missing. The number extracted only becomes important when the vast majority of the tree needs to be explored, as in learning for example.

The variance of the model used for recognition is also important, since with a large variance the A* search heuristic will be less useful. This means more of the search tree will be expanded, making it less efficient. The variance of the hand-trained model was artificially altered to investigate this.

The queue is implemented as a binary heap which has $\lg n$ efficiency for extract-max and insert operations. Some kind of quad-tree approaches were tried to provide efficient implementation of the space thresholding. However, given that we only use an image once, the overhead of creating such spatial data-structures, made it redundant. In cases where the images would be reused, in learning for example, this approach would prove advantageous.

5. Results

The 10 features used the recognition model were: Left Eye; Right eye; Left Mouth; Right Mouth; Centre Hairline; Right Hairline; Left Hairline; Nose; Left Chin; Right Chin. The model is shown in Fig.2, along with the d_f for each feature.

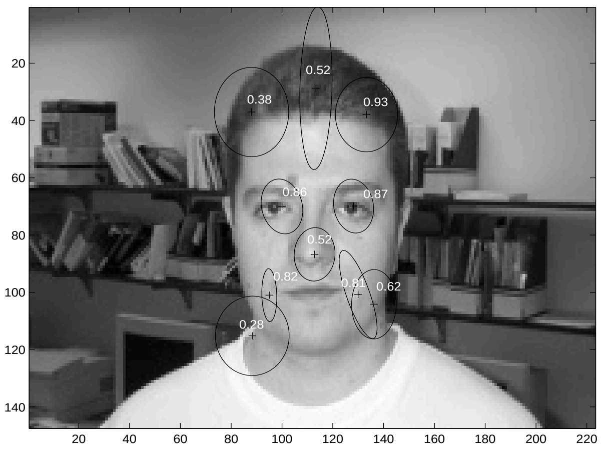


Figure 2: A face image, with a 10 feature model superimposed

An important test is how many features can be handled in a reasonable time. In Fig.3(a) with the average number of detections, a, equal to 5, we see that beyond 7 features, the exhaustive becomes too slow to be practical. The A* is still quick enough at 10 features for real-time recognition. Extrapolating, A* would take a second per image for 13 features.

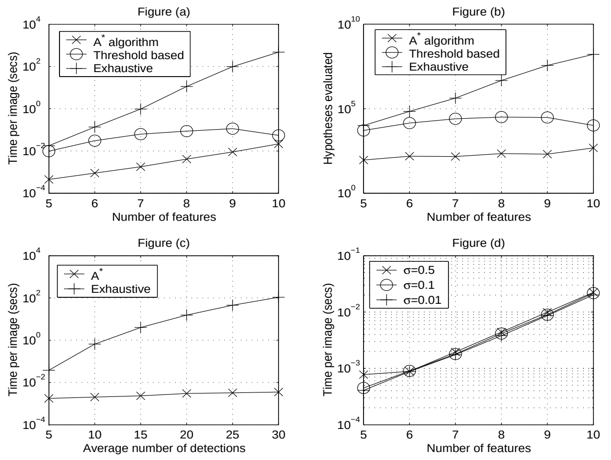


Figure 3: A comparison between the different search methods

The threshold approach gives some rather confusing results, but the curve seen is due to the upward relative movement of the threshold, as mentioned earlier. Relatively fewer hypotheses are being evaluated as F increases. Without setting the threshold at every value of F, it is not possible to remove this effect. Therefore we will no longer compare the threshold approach to the other two.

Fig.3(b) gives an idea of the exponential explosion of hypotheses. With 10 features, with $a = 5$, we have around 100 million hypotheses on average per image. This figure also illustrates, when compared to Fig. 3(c) that the time per hypothesis of the different techniques is fairly constant, roughly within a factor of 2 of each other.

Fig.3(c) shows how a effects the evaluation time, for $F = 5$. We would expect from $O(a^F)$, altering a is less detrimental than altering F. A practical demonstration of this is given in Fig. 5. It shows a 5 feature model locating a face in two composite images which have a large number of detections (a =35), which resulted in 5×10^7 hypotheses per collage. For each one, using the exhaustive search it took around 110 seconds to find the best hypothesis, but with A* it took 0.015 seconds. The space thresholding approach means performance scales well as a increases, so giving the flat curve.

Fig.3(d) demonstrates that the A* algorithm is not affected much by altering the variance of the model. Since the detection data was rescaled to be in the range [0:2.4,0:1.5], $\sigma = 0.5$ would cover most of the image.

Fig. 4 shows how, in the threshold approach, the number of hypotheses is reduced as the threshold increases. Since this curve will be shifted to the left or right considerably, depending on F and a, the task of choosing an appropriate threshold is a difficult one to do in advance.

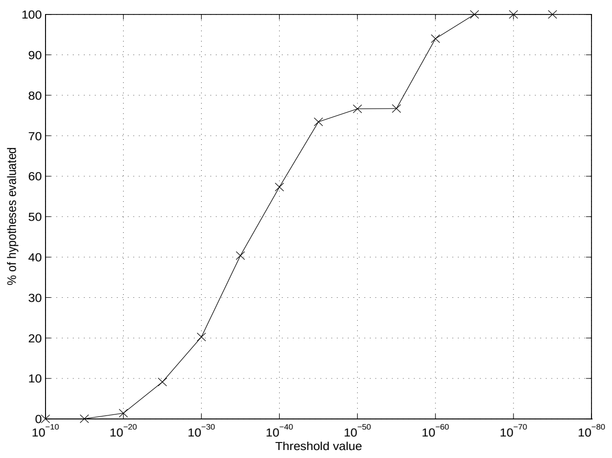
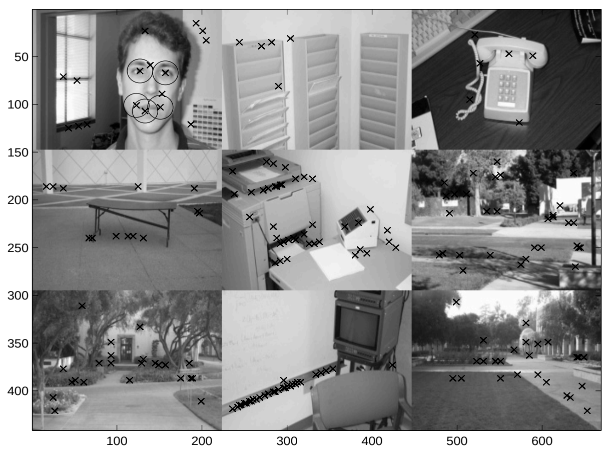


Figure 4: The effect of the threshold on the number of hypotheses considered.



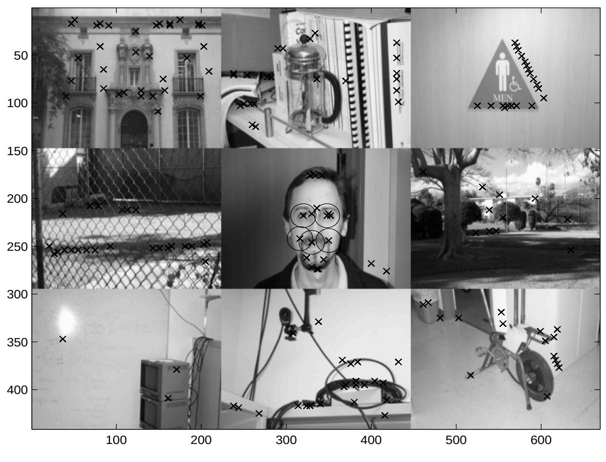


Figure 5: Two collages of background images and one face image. The crosses are feature detections. The best hypothesis is indicated by the circles.

6. Conclusions

We have presented two efficient methods to search for the best hypotheses. However, of the two, A* is the better.

Firstly, it has no tuning parameters as compared with the threshold method. As seen, the threshold must be chosen with care to avoid the method failing. A* is also provably correct in its basic form. The addition of the space thresholding, while a heuristic, does not cause the method to fail in practical situations.

The orders of magnitude speed improvement of A* over the exhaustive search allow us to do real-time detection on large images. Detection at 60Hz is possible with $F = 10$, $a = 5$ or with $F = 5$ and $a = 35$.

The spatial thresholding means A* scales well with a: with $a = 100$ and $F = 5$, it still took only 1 second to find the best hypothesis. This means we can deal with the multi-mega-pixel images of modern digital cameras.

The increased range in F is less pronounced, but it is worth remembering that a 10 feature model would require a very large number $O(10^{3-4})$ of images to train properly (to prevent over-fitting), making them impractical to train. Therefore, $F = 10$ is a sensible practical maximum for F.

While it has not been discussed here, the A* and space thresholding methods can be applied to the related problem of automatic model learning. Learning is a harder problem, since at the beginning, we know nothing about any of the hypotheses, so we cannot ignore any of them. In the latter stages it should be possible to discard most of the hypotheses, but a principled approach is required.

References

[1] M.A. Fischler and R.A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computer*, vol. c-22, no. 1, pp. 67–92, Jan. 1973.

[2] T.F. Cootes and C.J. Taylor, ""Locating Objects of Varying Shape Using Statistical Feature Detectors"," in *European Conf. on Computer Vision*, 1996, pp. 465–474.

[3] M. Lades, J.C. Vorbruggen, J. Buhmann, J. Lange, C. v.d. Malsburg, R.P. Wurtz, and W. Konen, "Distortion invariant object recognition in the dynamic link architecture," *IEEE Trans. Comput.*, vol. 42, no. 3, pp. 300–311, Mar 1993.

[4] A.L. Yuille, "Deformable templates for face recognition," *J. of Cognitive Neurosci.*, vol. 3, no. 1, pp. 59–70, 1991.

[5] M.C. Burl, T.K. Leung, and P. Perona, "Face localization via shape statistics," in *Int. Workshop on Automatic Face and Gesture Recognition*, 1995.

[6] M.C. Burl, T.K. Leung, and P. Perona, "A probabilistic approach to object recognition using local photometry and global geometry," in *Proc. 5th Europ. Conf. Comput. Vision, H. Burkhardt and B. Neumann (Eds.), LNCS-Series Vol. 1406–1407, Springer-Verlag*, 1996.

[7] M. Weber, M. Welling, and P. Perona, "Towards automatic discovery of object categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, June 2000.

[8] M. Weber, M. Welling, and P. Perona, "Unsupervised learning of models for recognition," in *Proc. 6th Europ. Conf. Comp. Vis., ECCV2000*, June 2000.

[9] M.C. Burl, T.K. Leung, and P. Perona, "Recognition of planar object classes," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, 1996.

[10] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical society* **B**, vol. 39, pp. 1–38, 1976.

[11] M. Weber, *Unsupervised Learning of Models for Object Recognition*, Ph.D. thesis, Department of Computation and Neural Systems, California Institute of Technology, Pasadena, CA, 2000.

[12] P.E. Hart, N.J. Nilsson, and B. Raphael, "A formal basis for the determination of minimum cost paths," *IEEE Transactions on SSC*, vol. 4, pp. 100–107, 1968.