Proceedings

Performance versus Methodology

in

Computer Vision

NSF/ARPA Workshop

June 24–25, 1994

Seattle, Washington, USA

in cooperaration with

IEEE Computer Society

These papers comprise the proceedings of the meeting mentioned on the cover and on the title page. They reflect the authors' opinions and are published as presented and without change, in the interests of timely dissemination. Their inclusion in the publication does not necessarily constitute endorsement by the editors or the Intitute of Electrical and Electronics Engineers.

Additional copies may be ordered from:

Marlene J. Feinman
Workshop: Performance vs. Methodology
in Computer Vision
Department of Electrical Engineering, FT-10
University of Washington
Seattle, WA 98195 USA

Matched Filtering and Bayesian Pattern Matching

Robert A. Hummel
Courant Institute
New York University
New York, NY, 10012

Abstract

We begin with some polemics about computer vision research and pattern matching, and the need for robustness. We view a large portion of computer vision as matched filtering, suitably embellished, often disguised. This broad statement is justified with a number of examples. Typically, matched filtering arises when problems are formulated as L^2 optimization problems. We then consider a formulation of pattern matching as a Bayesian maximum likelihood computation, as opposed to a least square norm minimization. In this formulation, we discover that certain weighted functions that would ordinarily be invoked heuristically in order to account for noise and statistical variation can be derived and given precise meaning. We argue that the formulas that are derived has heuristically the right behavior, but will provide better performance than other heuristic formulas when applied to real applications.

1 Introduction

Computer vision research is beginning to show promise for commercialization. Small demonstration projects in research labs, particularly in robotics, manufacturing, and parts inspection, demonstrate capabilities that seemed unlikely a decade ago (and yet were promised to be around the corner two decades ago). Aerospace contractors are beginning to develop production automatic target recognition systems, and renewed interest in countering armored threats in regional conflicts has renewed research funding in image processing for target recognition.

This time around, the output had better be more than a bunch of conference and journal articles. It is time for the performance of the vision systems to provide realistic, commercializable applications, and that the performance of systems be quantifiably related to the parameters, scenarios, and inputs (e.g., the sensors) in the system. The demonstrations must be extensible, and the benefits must be quantifiable. Marketing, support, modifications, and product life-cycles will be new challenges in computer vision applications. Those who recall the demise of Machine Vision International and other startups in past decades might argue that such concerns will be renewed challenges, but if the products are actually useful this time, then the challenges will be far more complicated and intricate.

There is good reason to believe that narrowly-defined applications with sizable markets are finally amenable to computer vision technology. It is not so much that the techniques developed over the years have reached fruition, but rather that processor speeds, DSP technology, sensors, and reduced prices make the economics of simple techniques more viable. A potential difficulty is that the customer with the deepest pockets, the military, is currently in need of cost-containment, so that procurement of real vision systems might be slowed after development and demonstration, but if demonstrations are truly satisfactory and extensible, then the "dual-use" market will include medical, manufacturing, navigation and collision avoidance, and other applications. Further, the price reductions have been dramatic. Many researchers still have $50,000 to $100,000 attached image processing systems in storage, in favor of $5,000 SCSI image digitizers, and others are beginning to look at the $500 camera systems that can be attached to workstations, with the intention that demonstrations system use commonly available inexpensive hardware.

Are we prepared? Alas, academic researchers have little appreciation of the subtleties of the marketplace, let alone the range of applications and requirements of the customers. Academics, and even industrial research labs, know mostly about impressing other academics and researchers, and have developed a broad array of methodologies for this purpose. In the new world, where DSP chips can easily achieve 700+ MIPS, the goal is not sophisticated algorithms nor large numbers of published journal articles nor deep theorems that invoke obscure mathematics, but rather simple systems that work well and fail gracefully and are easily adapted to new environments.

The methodologies that researchers utilize in order to develop demonstrations must be modified to account for the new goals. In particular, it is critical that the performance of a system be demonstrated over a range of potential inputs, and that robustness of a system form a critical component of its evaluation. Indeed, the ideal journal article should now begin with results, proceed by demonstrating extensibility of the results, document performance statistically, and end with a short section demonstrating the simplicity of the algorithm.

The remainder of this paper is mostly technical. The comments that follow argue generally for the use of Bayesian methods, in the context of pattern matching, for model-based vision applications. We assert that Bayesian methods lead to well-justified formulas, and reasonable formulations. The principle alternative, which has been a mainstay of computer vision algorithm development, is based on optimization, and generally leads to some form of matched filtering. We begin with a discussion of classical matched filtering.

2 Matched Filtering

By matched filtering, we mean simply an inner product. So if $f(x, y)$ is an image, and $m(x, y)$ is a model or pattern that is being sought, we compute

$$\int f(x,y) \cdot m(x,y) dx dy$$

in order to determine the degree of match. If multiple patterns are sought, say $m_i(x, y)$, for $i = 1, \cdots n$, then n inner products are computed, and the largest wins.

Beginning with optical character recognition, and continuing with optical computing and target recognition, matched filtering holds an important appeal and promise. The technique is easily implemented, and has mathematical justification. Specifically, if one wants to find the best match in terms of the L^2 norm, then the minimum of $||f - m_i||^2$ is equivalent to maximizing $\langle f, m_i\rangle - (1/2)||m_i||^2$. If all the models have the same norm, then we have derived matched filtering, and if the models have different norms, then we have good justification for a minor modification to matched filtering.

Alas, it is doctrine in computer vision, and to a lesser extent in pattern recognition, that matched filtering does not work. Unless the number of models is minimal, noise and variability kills the technique. The doctrine is re-learned by successive generation of researchers, and is verified empirically and by theoretical analysis.

But, as we will see, variations of matched filtering reappear in many guises. The methodology of computer vision research seems to dictate that matched filtering form the key component of the matching engine, providing it is sufficiently embellished. The performance of these systems, as we all well know, is adequate for the examples that are published with the papers, but falls apart when the method is extended to more models or more complicated systems. The problem of dealing with noise remains the key difficulty in the application of computer vision technology. The source of the problem is that matched filtering can't deal with noise effectively, which has been known for decades.

At the same time, Bayesian techniques show great promise in alleviating problems with noise and inadequate matching engines. The difficulty with Bayesian techniques lie in the representation of the information that is to be analyzed. The representation is critical, but is not dictated by any theory. Accordingly, Bayesian networks or Bayesian reasoning systems have to be developed for each application, often in ad-hoc ways. A better methodology is required.

In the sections to follow, we will not solve the problem of formulation the representation in order to utilize Bayesian reasoning to break the log-jam of matched filtering. However, we will show how matched filtering itself can be reformulated and modified so as to admit a Bayesian interpretation, which should assist providing new methodologies that lead to more robust performance.

3 Other guises for matched filtering

Matched filtering can be formulated as the task of maximizing a collection of vector dot products. If $\vec{x}$ is a vector representing the ordered pixel values, and $\{\vec{x}_i\}_{i=1}^n$ is a collection of target vectors, then the problem is to find the index that maximizes $\vec{x} \cdot \vec{x}_i$ over all possible i. Since the $\vec{x}_i$ will typically contain multiple translates of the same prototype pattern, many of the vector dot products can be efficiently implemented as convolutions. Since the $\vec{x}_i$ are not orthogonal, there can be considerable cross-talk. The vector $\vec{x}$ of pixel values may represent the result of an edge detector or other filter of raw sensor data, and may also incorporate multispectral and multisensor information.

Next, let us consider statistical pattern recognition. Although there are many forms, let us first consider the K-nearest neighbor classifier, with K equal to one. In this classifier, a feature vector $\vec{x}$ locates the prototype that is nearest, i.e., that minimizes $||\vec{x} - \vec{x}_i||^2$ over i. This is equivalent to maximizing

$$\vec{x} \cdot \vec{x}_i - \frac{1}{2}||\vec{x}_i||^2,$$

which we may write as $(\vec{x}, 1) \cdot (\vec{x}_i, b_i)$, where $b_i = -(1/2)||\vec{x}_i||^2$. Thus we once again get a vector dot product maximization, but in this case, the vector represents the feature values of a region of interest with an appended component that is always one, and the matched filters encode the prototype feature vectors, with an appended bias term.

What about classifiers with K greater than one? Perhaps there is something essential in better pattern recognition methods and more sophisticated classifiers that relieve them from the tyranny of matched filtering. For example, a two-class 5-nearest neighbor classifier might use a voting scheme among the five nearest neighbors to a test pattern, resulting in one of the two classes assigned to the test. However, the result of the voting scheme is that the multiparameter feature space is decomposed into regions that result in assignments to one class or another, and the resulting decomposition can be closely approximated by a 1-nearest neighbor classifier, providing enough prototypes are inserted. In other words, by changing the prototypes, a 1-nearest neighbor classifier can be used to approximate just about any static pattern recognition scheme.

4 Model-based vision

Next, let us consider a model-based vision application. Suppose that we are using an hypothesis-and-test approach, such as the Lowe SCERPO method [1] or successive methods. As some stage in the processing, there is a 3-D model m that is hypothesized to be present, and some number of parameters that have been estimated, so as to predict certain features. The system must determine other parameter values, and/or verify currently established values, updating parameters according to new features extracted from the image. There are two phases to this process: (1) matching scene features to model features that can be predicted in the image, but may be located along a multiparameter transformation orbit; (2) improving viewpoint parameters based on newly discovered feature matches.

The first of these two phases, matching scene features to model features, can be formulated as follows. We view scene features as vectors $\vec{y}_i$, which include position information in the scene, and can also include attribute information. Likewise, the hypothesized model is composed of a similar collection of features, say $\vec{z}_j$. We have a transformation T that depends on unknown parameter values $p_1 \cdots p_n$, we wish to determine parameter values such that the collection of transformed model features,

$$\{T(p_1 \cdots p_n; \vec{z}_j)\}$$

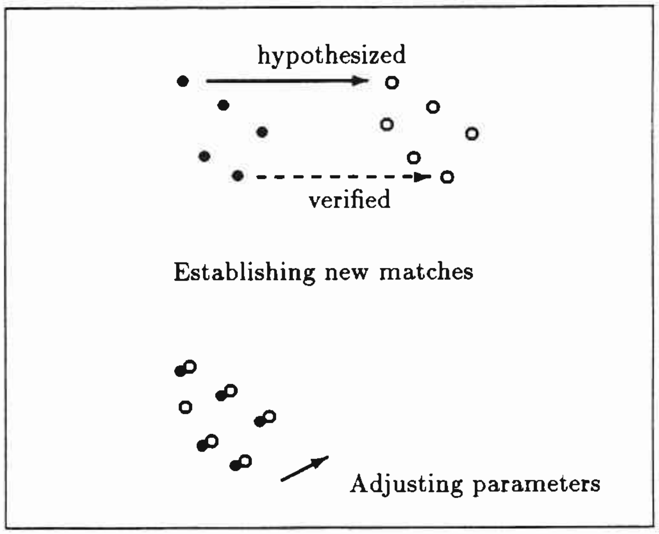
best approximates a subset of observed scene features

$$\{\vec{y}_i\}.$$

A metric is required in order to measure the distance, and a typical measure is a mean square metric in feature space. This metric is common due to its mathematical tractability—the parameter search is facilitated when the minimization can be performed assuming an L^2 metric. Nonetheless, the problem has many subtleties, which is familiar to researchers in computer vision as the *correspondence problem*. In various forms, the problem can be solved by subgraph isomorphism, relational graph matching, relaxation methods, analytic optimization, and by many other methods. In general, some of the parameter values may be fixed, others constrained, and others are completely undetermined, and certain correspondences of features may already be established, and new correspondences are sought. The metric measuring the degree of match may well be changed as matches are hypothesized. However, in any given iteration, new matches are sought, either along with providing additional constraints on parameters, or as a verification to already determined parameters, and the metric to evaluate correspondences is fixed for the iteration. Thus, if all p_i are free, then we wish to find a subset S of the observed features, a map from the subset S matching $\vec{y}_i \in S$ to a subset of the model features $\vec{z}_{j_i}$, such that the minimum

$$\min_{p_1 \cdots p_n} \sum_{\vec{z}_i \in S} \|T(p_1 \cdots p_n; \vec{z}_{j_i}) - \vec{y}_{j_i}\|^2$$

is small relative to the number of matches in S. One formulation establishes a threshold fraction for each model, and declares a model to be recognized if there exists a matching of at least the specified fraction of model features to features in the scene such that the minimum parameter fit error (above) gives a sufficiently small value. Other ways of formulating minimization problems with constraints, including penalization methods, are discussed in [2].



There are many special cases, some of which we discuss later.

The second phase, viewpoint refinement, is formulated as follows. We have a collection of scene features $\vec{y}_i$, for $i = 1 \ldots k$, in correspondence with specific features in a hypothesized model, say $\vec{z}_{j_i}$, $i = 1 \ldots k$. Using a norm that can take into account the stated correspondences, we wish to find the optimal parameters $p_1 \cdots p_n$ in order to minimize the distance from

$$\{T(p_1 \cdots p_n; \vec{z}_{j_i})\}_{i=1}^{k}$$

to the set of features $\{\vec{y}_i\}_{i=1}^{k}$. In a sense, the second phase is the same as the first, except that the collection of features are now fixed and in correspondence, whereas the matching phase described above must posit correspondences as part of the solution.

Suppose we fix on L^2 norm. Then the problem is to find the $p_1 \cdots p_n$ minimizing

$$\sum_{i=1}^{k} \|T(p_1 \cdots p_n; \vec{z}_{j_i}) - \vec{y}_{j_i}\|^2.$$

The figure depicts the two problems: finding matches and parameter refinement.

Clearly, both phases are optimization problems, and can be solved by many different approaches to numerical optimization.

Suppose that T is linear in all its variables. This is a special case that is unrealistic, but nonetheless surprisingly common, and will be approximately true much of the time in any case. Clearly, a local linear approximation is often valid, and since iterative optimization operates locally, a linear assumption is not totally unuseful.

In that case, the solution to both phases, for a fixed model and fixed matching, is a linear minimization problem, of the form

$$\text{minimize } \|Ax - \vec{b}\|^2 \text{ over } x$$

whose solution is given by the solution to the "normal equation," and can be solved by a simple iterative procedures; equivalently the minimum value can be obtained by evaluating $||[A(A'A)^{-1}A' - I]\vec{b}||^2$. Since $\vec{b}$ alone depends on the extracted scene features $\{\vec{y}_i\}$, and A depends on the current parameter values and the model features $\{\vec{z}_j\}$, the minimization is implementable as a vector dot product (i.e., a matched filter operation), such that the minimum value can be formulated as a vector product $\vec{x} \cdot \vec{x}_i$, where $\vec{x}$ depends on the values in the matrix A, and hence on the model features $\vec{z}_j$, and $\vec{x}_i$ depends on the vector $\vec{b}$, and hence on the $\vec{y}_i$'s, although the functional relation may not necessarily be simple.

Of course, for the first phase, the model parameters, the subset of model features, and the match to scene features are not fixed. Thus many innovative search strategies can be developed to perform the minimization in an efficient manner. However, here's a brute force method. We begin by enumerating all possible matches, over all possible subsets of model features, over all possible models, which we index by α. Further, for each α, some of the model parameters are fixed, and the others are free. For each such α, the minimization is a linear problem, as describe above, and leads to a vector product $\vec{x} \cdot \vec{x}_\alpha$ to obtain the value to be minimized (or maximized). If we find extrema over the many α, the result is posited recognitions.

For the second phase, no maximization is required over multiple α. Instead, a simple optimization is required. can be solved by matched filtering, or some related technique, depending on the metric norms. In the case of the first problem, since the parameter values are unknown, the set of available parameters might have to be discretized, which can lead to errors. Indeed, the correspondence problem is interesting precisely because one wants to find a search strategy that is better than exhaustive search.

We see that at least one form of the hypothesis-and-test approach to model-based vision can be implemented as a sequence of matched filtering decisions. Of course, this formulation made a linear assumption, or at least a local linear assumption, of the transformation of features T, and the approach that has been outlined is brute-force, and likely to be improved in actual implementation. Nonetheless, our formulation shows the pervasiveness of matched filtering.

5 Unstable problems and unstable methods

At this point, we have said that:

- Matched filtering won't work; and
- All methods lead to matched filtering, perhaps in disguise.

This theme will continue, even as we discuss geometric hashing.

There is a seeming dilemma. If we really believe the points, then we should give up hope. Either, matched filtering is not so bad after all, as long as we get the representation right, or matched filtering is the wrong approach, and we must make sure that the methods that are used to solve the problems are different than matched filtering. After all, we have only shown that standard vision problems have formulations amenable to solution by matched filtering. We have not said that matched filtering is the *only* approach to solving vision problems.

The point is that problems, particularly optimization problems, can be unstable or stable, and if they are stable, it is still possible to fail to solve them due to the fact that the methods are unstable.

The computer vision community hopes that the following is true:

1. The computer vision problems as formulated above are stable problems;
2. The decades of failure of matched filtering approaches means that matched filtering is an unstable method for solving most of these problems; and
3. Stable approaches will soon be found, or have been found, and are yet to be fully promulgated.

But it could also be that the problems as formulated are unstable, and that a better formulation is required. At this point, it is customary to point to biological vision systems as an existence proof, and thus support for the proposition that the recognition problem must be stable. Thus, the implication is that the fault lies in the methods, although it could be that the fault lies in the formulation. In particular, the optimization formulation is not necessarily the only way to describe the matching problem.

6 Geometric Hashing

Geometric hashing [3] is a method for organizing pattern matching, and is related to the object matching formulation given in the previous section. However, by discretizing the space of parameter values for the transformations in a clever way, the enumeration of possible matches becomes more manageable. Geometric hashing as a search method is particularly attractive because (1) it is parallelizable, (2) efficient especially when dealing with large databases, and (3) permits easy adaptation.

Using independent features with attributes, an image scene is represented using a collection of vectors, $\{\vec{y}_i\}$, as before. Likewise, each model is represented as a collection of features, say for model m_k, the features are $\{\vec{z}_{k,i}\}_{i=1}^{n_k}$. The recognition should be independent to some class of transformations that can be applied to the models, such as translation, rotation, and perhaps skew transformations.

Accordingly, we define a ***basis set*** to be a minimal collection of features such that placing a basis set in one-to-one correspondence with another basis set (such as a basis set in the scene) determines a transformation in the class of transformations under which recognition must be invariant. For example, if the features are points in $\Re^2$, and if recognition is to be similarity (translation, rotation, and scale) invariant, then a pair of points establishes a basis.

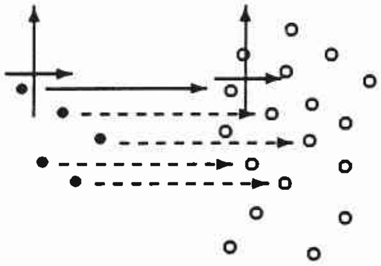


Figure 1: In the pattern matching problem, the origin is fixed in the pattern and in the scene, and we simply wish to observe the designated pattern in the scene.

For any particular basis, remaining features may be normalized with respect to the basis. So, if we choose a basis B in a model m_k, then all features $\vec{z}_{k,i}$ in m_k may be transformed in such a way as to move the basis to some pre-determined configuration. Likewise, given a basis B' in the scene, then all features $\vec{y}_i$ may be transformed so as to move the basis B' to the same pre-determined configuration. If model m_k occurs in the scene with basis B in the model in correspondence with the features B' in the scene, then after normalization, the normalized model features should occur within the normalized scene features in the same configuration.

Accordingly, rather than searching for transformation parameters by a numerical scheme, in conformance with the idea of enumerating over all possible matches indexed by α (as formulated in the previous section), we begin by expanding the model base, considering each model m_k and every basis B within m_k to be a prototype *pattern*. Thus the new collection of models, now called patterns, are pairs (m_k, B), subject to the condition that the basis B is a basis set of features in the model m_k.

To perform the search, we choose a basis B' in the scene, and normalize the features $\vec{y}_i$ with respect to the basis B'. The goal is to determine if any of the model/basis pairs (m_k, B) match the normalized scene. The question is the same as asking whether any of the models are embedded in the scene in such a way that the scene basis B' matches to any basis in the models. If all bases have been included in the enumeration, then it suffices that B' simply lies within an embedded model. The match problem is depicted in the figure.

For any particular model/basis (m_k, B), the question is very simple. We have a collection of normalized model features, which we call $\{\vec{x}_i\}$. We have a collection of normalized scene features $\{\vec{u}_j\}$. We want to find an embedding of $i \rightarrow j_i$ for a subset of the possible i such that each $\vec{x}_i$ lies "near" $\vec{u}_{j_i}$.

This problem can be formulated as an optimization problem, but let us consider a heuristic matched filtering approach directly. After all, we are attempting to find a match of the pattern $\{\vec{x}_i\}$ within $\{\vec{u}_j\}$. We now revert to a continuous formulation.

Let

$$f(\vec{x}) = \sum_i \delta(\vec{x} - \vec{x}_i),$$

and let

$$g(\vec{x}) = \sum_j \delta(\vec{x} - \vec{u}_j).$$

(Note that f is dependent on the model/basis pair (m_k, B), and so we might denote it as $f_{(m_k,B)}$. Nominally, the problem is to determine whether delta peaks in f match a subset of peaks in g. Heuristically, this can be determined by considering $\langle f, g \rangle$, although clearly some "smearing" of the peaks is required in order to account for normal noise and possible minor displacements. One way of doing this is to "blur" f by a gaussian,

$$\tilde{f} = f * G_C$$

, where G_C is a gaussian distribution with mean zero and covariance C. We then compute $\langle \tilde{f}, g \rangle$.

A better approach modifies f according to expected variations of the normalized features. Let us suppose that the normalized features $\vec{x}_i$ have expected covariance variations of C_i respectively, when found embedded in the typical (normalized) scene. Then, we redefine $\tilde{f}$ as

$$\tilde{f}(\vec{x}) = \sum_i G_{C_i}(\vec{x} - \vec{x}_i),$$

where G_{C_i} is a Gaussian with covariance C_i. Then the inner product $\langle \tilde{f}, g \rangle$ measures not only the number of features represented by peaks in $\tilde{f}$ that have matches in the scene represented by peaks in g, but also takes into account how closely the matches line up.

Clearly, we are back to matched filtering. Worse, the justification is heuristic. And it is not clear if the scheme will not work because it is matched filtering, or it won't work because the problem is still formulated poorly, or perhaps it will work if the features are robust, stable, and discriminative.

7 Bayesian Pattern Matching

This section shows that the fundamental pattern matching problem, formulated in the previous section, can be given a Bayesian formulation, leading to both a justification and a modification of the heuristically derived formula for matched-filtering -type recognition. Our optimistic conclusion is that matched filtering is probably okay after all, but that a non-optimization formula is needed for stability. Moreover, the right formula is critical, and the features and representation of the features is even more critical. This formulation is based on the thesis work of Isidore Rigoutsos [4].

The event that we hypothesize is that model m_k is present in the scene with basis B in m_k matching scene basis B'. We may denote this binary event as $E(m_k, B, B')$. A maximum likelihood formulation asks us to compute and maximize

$$\Pr(E(m_k, B, B') | \{\vec{u}_j\}),$$

where the $\vec{u}_j$ are the unnormalized scene features. Note that this formulation is slightly nonstandard, in that if the model m_k appears in the scene, then there may well be multiple pairs (B, B') for which the event is true, and which "discovers" the model. We only need find one such pair, if the model is present.

If the features are conditionally independent, then the above expression can be decomposed. The conditional independence assumption is a strong one, and imposes a large constraint on the kinds of features with which we can operate. The condition means that under the assumption that a particular model is present in a particular location, knowledge of a feature at one location (or with one set of parameters) has no influence on the probability density distribution of another feature. The condition is not at all true, for example, when dealing with edge elements (edgels), which tend to line up in lines, and are thus very much dependent.

Using various Bayesian manipulations, and using the independence assumption, one can derive

$$\log\left(\frac{\Pr(E(m_k, B, B'))|\{\vec{u}_j\}}{\Pr(E(m_k, B, B'))}\right) =$$

$$\log K + \sum_j \log\left(\frac{\Pr(\vec{u}_j | E(m_k, B, B'))}{\Pr(\vec{u}_j)}\right)$$

where the probability terms in the sum should be interpreted as density function evaluations. See [5] for details of these manipulations. If we assume that all prior probabilities of the events $E(m_k, B, B')$ are uniform, then we can simply strive to maximize the right hand side. The $\log(K)$ term will be independent of (m_k, B, B'), and so can be ignored.

Accordingly, let us examine carefully the summand terms on the right hand side of the equation above in light of the patterns $\{\vec{x}_i\}$ and $\{\vec{u}_j\}$.

The denominator $\Pr(\vec{u}_j)$ is very simple. This is simply the prior probability of a feature occurring with values $\vec{u}_j$, where the features are chosen randomly in a random scene. There is a background density distribution, say $\rho(\vec{x})$ on the feature space, and this probability is simply an evaluation of this density function. This function can be derived from a modeling of the image space and of the feature extraction process, or could be developed empirically.

The numerator is a more complicated story. This density function is different, because it is conditioned on the knowledge that event $E(m_k, B, B')$ is true. This means that we know that the normalized scene contains the features $\{\vec{x}_i\}$, for $i = 1 \ldots n_k$, where there are n_k features in the model m_k. Let us suppose that we have s observed features. Let us further suppose that of the n_k features in the model m_k, it is reasonable to expect on average a certain amount of obscuration, and thus only $\beta \cdot n_k$ features will be observed from the model, and the remaining will be background clutter features. Finally, let us assume that each of the expected features $\vec{x}_i$ has an expected variation with covariance matrix C_i. Then the expected density distribution is given by

$$\tilde{\rho}(\vec{x}) = \frac{s - \beta n_k}{s} \cdot \rho(\vec{x}) + \frac{\beta}{s} \cdot \sum_{i=1}^{n_k} G_{C_i}(\vec{x} - \vec{x}_i).$$

Note that the function has total density one (assuming ρ has total density one), and that the delta masses at the locations of the pattern of the normalized model m_k have become Gaussian "bumps," as heuristically considered in the previous section. Accordingly, each summand has the form

$$\log\left(1 - \frac{\beta \cdot n_k}{s} + \frac{\beta}{s\rho(\vec{x})} \sum_i G_{C_i}(\vec{x} - \vec{x}_i)\right).$$

Plugging into the Bayesian formulation above, we see that the maximization of the likelihood is tantamount to computing, for each collection of $\{\vec{x}_i\}$ based on the model m_k and basis B, given the normalized scene features $\{\vec{u}_j\}$ based on the scene basis B',

$$\sum_{j=1}^{s} \log\left(\frac{s - \beta n_k}{s} + \frac{\beta}{s\rho(\vec{u}_j)} \sum_{i=1}^{n_k} G_{C_i}(\vec{u}_j - \vec{x}_i)\right).$$

This can be rewritten as

$$-s \log\left(\frac{s}{s - \beta n_k}\right) +$$

$$\sum_{j=1}^{s} \log\left(1 + \frac{\beta/\rho(\vec{x})}{s - \beta n_k} \sum_{i=1}^{n_k} G_{C_i}(\vec{u}_j - \vec{x}_i)\right).$$

And now for some magic. For any given $\vec{u}_j$ value, at most one of the values of $\vec{x}_i$ will lie close. Let's call the closest one i_j. Then in the second term, the Gaussian terms will all be essentially zero, except possibly for the term $G_{C_{i_j}}(\vec{x}_{i_j} - \vec{u}_j)$. Thus we may replace the sum over i with the single term i_j!

So we obtain the following formula. We fix B' in the scene, and obtain the collection $\{\vec{u}_j\}_{j=1}^{s}$. We have many prestored model/basis patterns, each one consisting of a collection of the form $\{\vec{x}_i\}_{i=1}^{n_k}$. The support level for a particular model/basis depends on finding the nearest model normalized feature $\vec{x}_{i_j}$ to each normalized scene feature $\vec{u}_j$. The support for the particular model/basis is then

$$-s \log\left(\frac{s}{s - \beta n_k}\right) +$$

$$\sum_{j=1}^{s} \log\left(1 + \frac{\beta/\rho(\vec{u}_j)}{s - \beta n_k} G_{C_{i_j}}(\vec{u}_j - \vec{x}_{i_j})\right).$$

The bias term $-s\log(s/(s - \beta n_k))$, which we will denote by c_k, depends only on the model m_k, and not on the basis B chosen within model m_k. Accordingly, the support may be rewritten as

$$c_k + \langle \tilde{f}_{(m_k,B)}, g \rangle,$$

where g is a sum of delta functions at the locations $\{\vec{u}_j\}_{j=1}^{s}$, as before, and $\tilde{f}_{(m_k,B)}$ is again redefined, and discussed briefly below.

But, at this point, we may note that the pattern matching problem has once again been formulated at matched filtering! The collection of biased inner products should be maximized over all (m_k, B), and then separate collections may be evaluated and maximized for different B' (which effects the function g, i.e., $g = g_{B'}$). Accordingly, if Bayesian pattern matching provides a more stable performance than optimization methodologies, it is not because matched filtering is inherently inappropriate. Rather, there is still hope that the Bayesian approach is better because the change in formulation provides different formulas and different representations.

Finally, we define the Bayesian pattern-based matched filter $\tilde{f}_{(m_k,B)}$. For a feature vector $\vec{x}$, and $i(\vec{x})$ defined as the index of the nearest normalized vector among $\{\vec{x}_i\}$, i.e., the collection of feature vectors in m_k normalized according to the basis B, we then have

$$\tilde{f}(\vec{x}) = \log\left(1 + \frac{\beta/\rho(\vec{x})}{s - \beta n_k} G_{C_{i(\vec{x})}}(\vec{x} - \vec{x}_{i(\vec{x})})\right).$$

By using fancy data structures, such as hash tables and k-D trees, or self-balancing trees, the computation of the multiple inner products can be made quite efficient. This is what we refer to as "geometric hashing." But it is not our purpose here to argue for indexing and/or hashing methods here. Instead, our point is that a Bayesian formulation leads to yet another matched filtering formula, but where the filter operates in a domain equivalent to the range space of a feature value (which we have viewed as a vector, because there can be multiple attributes to a single feature), and with a formula that is non-obvious and only somewhat intuitive. Indeed, the formula for $\tilde{f}$ above can be closely approximated by a sum of log terms (since if $\vec{x}$ is distant from a particular $\vec{x}_i$, then the Gaussian will essentially evaluate to zero, and the log term is thus also nearly zero), but this is slightly different than a sum of Gaussians, which was our heuristic formula from the previous section.

8 Comments

At this point, it would be typical and desirable to state that the revised formula derived in the previous section for Bayesian pattern matching yields much better results, which we can document with examples. However, if we did this, we would be guilty of exactly the kind of article criticized in the first section, where the results are given in a summary in the penultimate section. Moreover, the jury is still out. We believe that Bayesian pattern matching will work much better than traditional object recognition systems, given the same features, but this is simply an intuition.

But from the standpoint of methodologies, we have two points: (1) There are alternative formulations; (2) we still haven't gotten rid of matched filtering, although it is considerably transformed and disguised.

9 Summary and Conclusions

For many years, people have worked on feature extraction, with the conviction that stable, robust features are important. We now state here that stable, robust features are essential. Viewing recognition as a pattern matching problem, the patterns have to be representative of the objects, and the patterns are composed of extracted features. Our discussion above has been mostly about the matching engine. Although we have shown alternative methodologies and developed new matching formulas, because we always come back to matched filtering, it is entirely possible that the precise matching engine does not matter. We believe that it does matter, and that the Bayesian approach should lead to more robust performance. However, in no way does the alternative approach obviate the need for stable features. Indeed, if anything, since our Bayesian approach is based on matching patterns of extracted features, we have emphasized the need for stable features.

Acknowledgements

Thanks to Isidore Rigoutsos, with whom many of these ideas were formulated in the context of his thesis on Bayesian interpretations of geometric hashing.

References

[1] D. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, 1985.

[2] R. Hummel, B. Kimia, and S. Zucker, "Debluring Gaussian Blur," *Computer Vision, Graphics, and Image Processing* **38**, pp. 66-80, 1987.

[3] Y. Lamdan and H. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," In *Proceedings of the 2nd International Conference on Computer Vision*, pages 238–249, 1988.

[4] I. Rigoutsos, *Massively Parallel Bayesian Object Recognition*, Ph.D. thesis, New York University, August, 1992.

[5] E. Charniak and D. McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley, 1985.