A Class of One's Own

Image Classification Using Forward Recursion

Christopher Tsai and June Zhang

{ tsaic@stanford.edu, junez@stanford.edu }

Department of Electrical Engineering, Stanford University

***Abstract*—In order to sensibly label objects or regions in an image, computers often require foreknowledge of the classes or categories into which the objects must be divided. However, we consider *unsupervised learning* algorithms, in which the human user supplies only the *number* of classes – not the identity or the statistics of the classes themselves. Given a set of training images and feature extractors, the classification system we describe generates a set of classes that are maximally distinct in the feature domain we prescribe. We formulate the acquisition of statistical knowledge from the training data as a Gaussian parameter estimation problem with vector quantization, and we implement classification of test images in a forward recursion algorithm. Our analysis juxtaposes the relative importance of various design parameters, such as the direction in which we scan the image, the block size, the type of extracted features, and the number of classes we instruct the algorithm to find.**

***Index Terms*—image classification, image labeling, object recognition, statistical learning, unsupervised learning, Bayes-Gauss likelihood, vector quantization, parameter estimation, hidden Markov model, forward recursion**

I. Introduction

CLASSES qualitatively separate objects, features, or other recognizable structure in an image. A myriad of fields in artificial intelligence – robotic motion, autonomous vehicles, sensor networks, video surveillance, intelligent human-computer interaction, and artificial vision – can profit from intelligent object recognition, or, better yet, automatic categorization of pixels or blocks in an image.

Even without human specification of classes, structure can emerge from a classification algorithm. For example, a snapshot of nature often invites a number of logical classification schemes, each with its own underlying order. Land-and-water, ground-and-sky, and natural-and-manmade represent common binary classification schemes, but more specific categorization heuristics also exist. Perhaps an image contains numerous forms of foliage, such as grass, crops, weeds, bushes, and trees; or the image might feature a number of animals grazing in nature, each of a different species or appearance. However, even the most intuitive classification schemes that come most readily to mind often fail to capitalize on all the nuanced information available in an image. Perhaps the cars that we are trying to separate from the street differ by make and model, just as the animals we might group into one class might differ by species.

In these situations, unsupervised learning – automatic generation of class labels – can prove useful. As scientists, we might be curious if we can unearth and exploit any further structure in an image for classification, or if underlying structure exists beyond our initially perceived categories. In brief, we might want the order in an image to emerge on its own, without any prompting or prodding from a human observer.

II. The Hidden Markov Model Framework

A. Order of the Model

We model the class structure in an image with a hidden Markov model (HMM), which assumes dependence on the finite past. In a first-order Markov model, the class of each block in an image depends only on the class of the previous block. Therefore, as we accumulate statistics, we need determine only the transition probability from the previous block to the current block. Logically, this assumption holds because we make our blocks sufficiently small that those of the same class tend to cluster, making first-order dependence quite feasible.

B. Viable Forms of Adjacency

We must vectorize our inherently two-dimensional structure as a string of blocks (or features) so that we can exploit first-order dependence. As we determine how to traverse an image from block to block, several types of adjacency come to mind. We could concatenate rows of blocks, columns of blocks, or diagonals of the image:

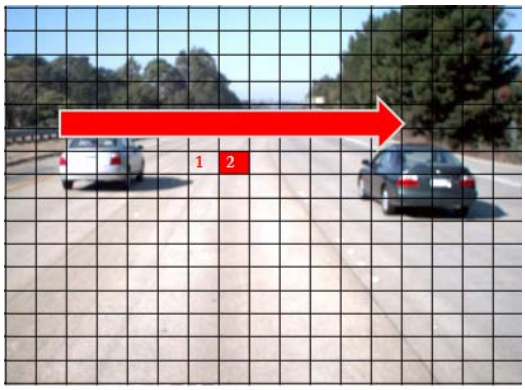

Fig. 1. Horizontal scan mode. The first (leftmost) element of each row depends only on the last (rightmost) element of the previous row.

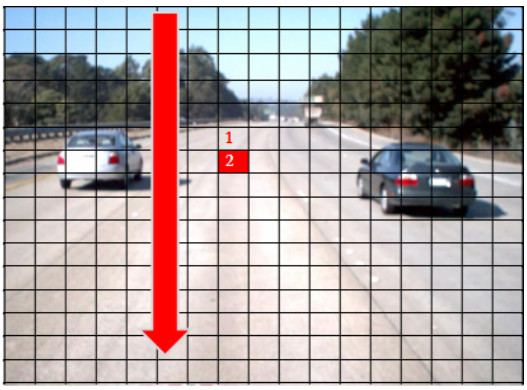

Fig. 2. Vertical scan mode. The first (uppermost) element of each column depends only on the last (lowermost) element of the previous column.

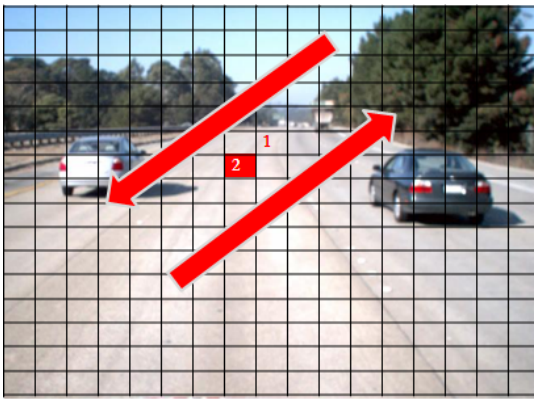

Fig. 3. Diagonal scan mode. Change direction at the end of a diagonal.

We can also envision horizontal and vertical scan modes that change direction at the end of a row or column instead of wrapping around. However, experimentation shows that, for images on the order of hundreds or thousands of pixels per row or column, the idiosyncrasies of wrapping around yield no consequential effect on the accuracy of classification.

C. The State Variable X_t as the Class

Our ultimate goal is blockwise classification of an image, so the variable of primary importance is the class label, which we make our Markov state X_t. For simplicity, we number the classes from 1 to k, where k represents the number of classes we instruct our algorithm to distinguish. True to the inherent mystery in the *hidden* Markov model assumption, the class of a block in a test image is never definitively known; it is a quantity that we must estimate. Even the human eye, trained to discern qualitative order, cannot penetrate the truth since the image is generally not physical reality but rather a noisy *representation* of reality. When given an image, we can work with only the observations.

D. The Observation Vector $\vec{Y}_t$ as the Features

Although the class of a particular block is not immediately obvious, the *features* that we choose to extract from the block are observables we can trust. Thus, we select a vector of numerical quantities that we deem our features vector or observation vector $\vec{Y}_t$. Altogether, the HMM model comprises hidden classes that manifest themselves through independent, noisy observation vectors $\vec{Y}_t$, as pictured below:

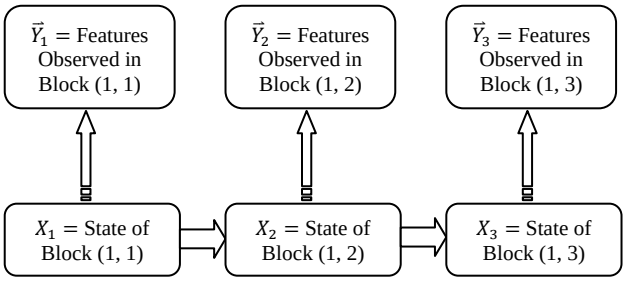

Fig. 4. First three states of the horizontal scan hidden Markov model.

Under the hidden Markov model with a finite number of possible states (classes), our system behaves in a predictable way. We can determine all transition and state probabilities given the observations using nonlinear techniques like forward recursion.

III. Classification System Overview and Outline

In this section, we delineate the classification system, treating each stage as a black box whose inputs we know and whose outputs we seek. The subsequent sections will open the boxes pictured below, stepping through our preferred algorithms.

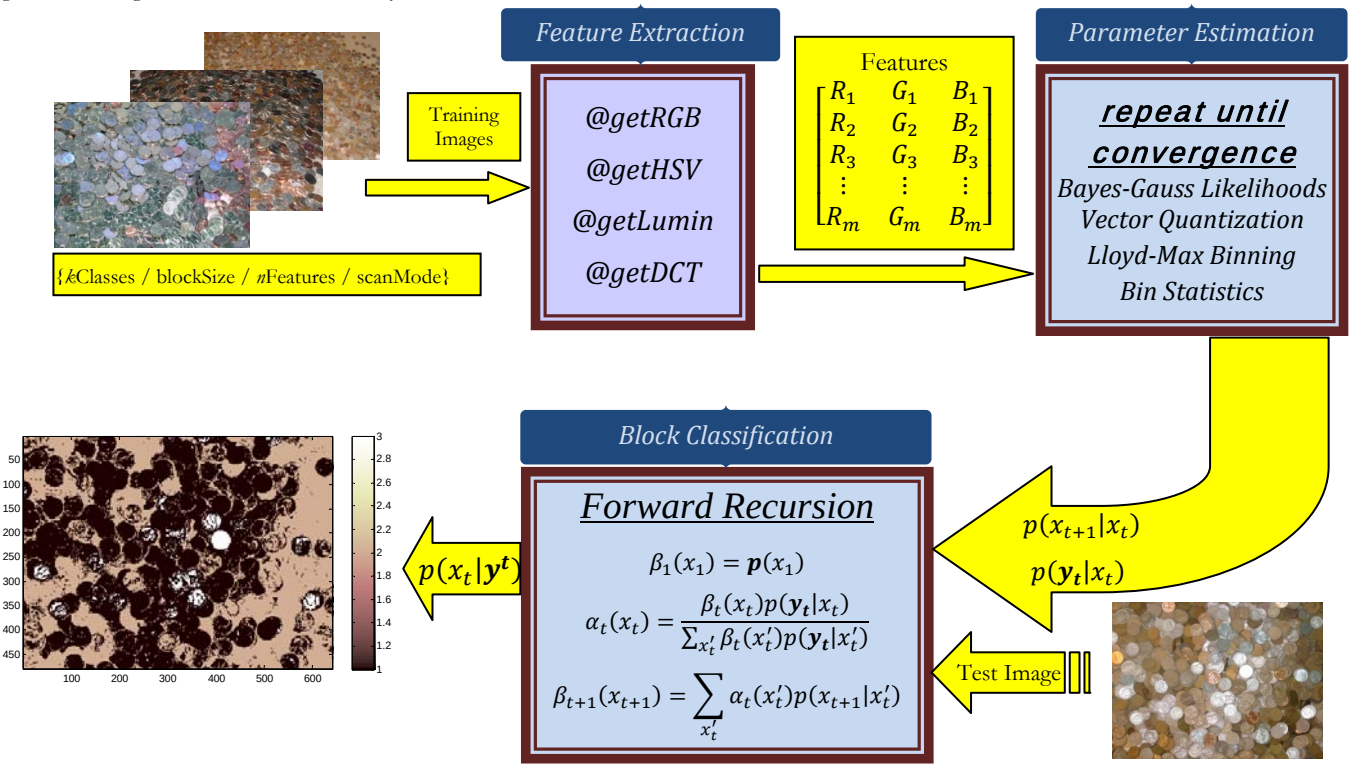

Fig. 5. Block diagram of classifying a test image based on parameters and probability distributions estimated from qualitatively similar training images.

A. Training Images and Design Parameters

Before we extract any features, we must determine the images from which we will extract them. We select a set of images that contain similar content to the test images that we wish to label; ideally, the similarity between training data and test data allows us to use the class-specific statistics that we generate from the training images in labeling the test image. In other words, the frequency of features in the training images will determine which classes we expect to see and separate in the test data, so it is vital to select training images that comprise a representative sample of objects we are likely to see in the test data. The number of training images that accurate classification demands will vary with the number of classes we choose to employ, since the incorporation of more classes will require more data to completely cover the larger number of possible transitions.

Our choice of design settings will also indirectly influence the result. Even though we might not know which classes will emerge from classification, we can limit the amount of variability in the model by setting the number of classes. Likewise, we select features that we think will best distinguish the different classes of data; for example, average color intensities might serve us well when labeling nature, but color descriptors fail for separating quarters from nickels and dimes, since all these coins share the same silver hue.

Fig. 6. Natural images display a range of vivid colors and respond well to RGB classification, whereas we need other features to separate silver coins.

The order in which our algorithm visits the blocks also perturbs the classification, but the differences are negligible.

Finally, based on the types of features we hope to extract, we select an adequate block size to achieve the necessary precision. For example, color features work even on pixel-size blocks, whereas edge detection, block transforms like the discrete cosine transform (DCT), and structuring elements require larger blocks for object structure to emerge.

B. Feature Extraction

Once we settle on a set of design parameters and training images, we extract the desired features from the set of training images. Block by block, our algorithm steps through each training image in the specified scan direction, applying the function pointer that we send into the feature extractor. The features from one training image are concatenated onto the features of the following training image, so only the transition from the last block of one image to the first block of the next image suffers from discontinuity.

Once the feature extractor has processed all blocks from all training images, the training images are no longer necessary, and our algorithm proceeds with a block-by-block features matrix. Each of the m rows in this matrix represents the feature or observations vector $\vec{Y}_t$ for a single block *t*. In other words, the i^{th} row of our features matrix is $\vec{Y}_i^T$.

Each of the n columns in this features matrix represents a particular feature. If we select the average color intensities in the red, green, and blue (RGB) channels as our feature descriptors, then our features matrix will contain three columns, as we will extract the three colors from each block.

Features Matrix

$$\begin{bmatrix} R_1 & G_1 & B_1 \\ R_2 & G_2 & B_2 \\ R_3 & G_3 & B_3 \\ \vdots & \vdots & \vdots \\ R_m & G_m & B_m \end{bmatrix}$$

C. Parameter Estimation

The features extracted from our training images will determine the classes toward which our classification algorithm converges. The goal of parameter estimation is to establish the probability density functions of these classes as a function of the feature vectors that we will extract from our test images.

In order to perform classification using forward recursion, we need two probability distributions:

$p(\boldsymbol{y_t}|x_t) =$ prob. of observing features $\boldsymbol{y_t}$ in a block of class x_t

$p(x_{t+1}|x_t) =$ prob. of block in class x_{t+1} after block in class x_t

Assuming a Gaussian mixture model, we treat the distribution of observable feature vectors in a block of class x_t as a continuous Gaussian density function, continuous in multivariate feature space $\boldsymbol{y_t}$. This mixture of Gaussians – one for each class – allows us to employ analogous classification functions, each expressing the likelihood of an observable set of features falling into its respective class. Tapping iterative vector quantization, we repeatedly classify all features until the classification statistics defining the Gaussian likelihoods converge. These statistics then allow us to express the desired probability density:

$$p(\boldsymbol{y_t}|x_t = class_i) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_i|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(\boldsymbol{y_t}-\boldsymbol{\mu_i})^T\Sigma_i^{-1}(\boldsymbol{y_t}-\boldsymbol{\mu_i})}$$

D. Forward Recursion

Armed with both the transition probabilities and the observation distributions, we are prepared to enter forward recursion with the test image(s) we hope to classify. Forward recursion relies on a step-by-step iteration involving known quantities to compute the distribution of $p(x_t|\boldsymbol{y^t})$ for each block *t*. This conditional distribution expresses the probability of the block's belonging to each class x_t given the observations or features $\boldsymbol{y^t}$ that we extract from the adjacent blocks. The forward recursion algorithm proceeds as follows:

$$\beta_1(x_1) = p(x_1)$$

$$\alpha_t(x_t) = \frac{\beta_t(x_t)p(\boldsymbol{y_t}|x_t)}{\sum_{x_t'}\beta_t(x_t')p(\boldsymbol{y_t}|x_t')}$$

$$\beta_{t+1}(x_{t+1}) = \sum_{x_t'}\alpha_t(x_t')p(x_{t+1}|x_t')$$

The coveted probability distribution is $\alpha_t(x_t) \triangleq p(x_t|\boldsymbol{y^t})$.

E. Classification: Quantizing the Distribution Vector

The distribution derived in forward recursion is a sequence of probabilities for each block *t*. Since our algorithm must definitively classify every block by a single class label, the final step involves assigning a single class label to each block. To resolve the sequence of probabilities in $p(x_t|\boldsymbol{y^t})$, we assume the block to belong to the class with the largest probability; we assign the class for which the value of $p(x_t|\boldsymbol{y^t})$ is largest for the block.

IV. Parameter Estimation using Vector Quantization

First, we assume a Gaussian mixture model, in that every feature belongs to a class whose distribution of feature vectors is Gaussian. This assumption holds not only mathematically – as information theory and/or the Central Limit Theorem can divulge **[5][6]** – but also intuitively, if we consider the behavior of image capture. Inevitably, noise from image capture corrupts and slightly perturbs the true nature of the imaged phenomena, but the inherent features of a certain class – stripes on a zebra, blue in the sky, and copper on a penny – persist and firmly establish the mean behavior that one expects to see. A combination of uneven illumination, motion blur, camera imperfections, and image compression degrade the pristine feature vector so that the distribution we witness from our training images assumes an approximately Gaussian shape, allowing us to model the distribution of observations or features as a Gaussian with class-dependent statistics:

$$p(\boldsymbol{y_t}|x_t = class_i) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_i|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(\boldsymbol{y_t}-\boldsymbol{\mu_i})^T\Sigma_i^{-1}(\boldsymbol{y_t}-\boldsymbol{\mu_i})}$$

where *n* is the number of features in $\boldsymbol{y_t}$. Under this model, all we need to ascertain $p(\boldsymbol{y_t}|x_t = class_i)$ are the mean and covariance statistics pertaining to each class-specific Gaussian. An iterative parameter estimation algorithm like expectation maximization suffices, but we choose vector quantization with Gaussian decision functions to parallel our assumption of a Gaussian mixture model for $p(\boldsymbol{y_t}|x_t = class_i)$.

Suppose we limit classification to k labels. For each of our m blocks, we evaluate each of k Gaussian likelihood functions, whose statistics $\{\boldsymbol{\mu_i}, \Sigma_i, p_i\}$ we initialize based on various heuristics beyond the scope of this discussion:

$$d_1(\boldsymbol{y_t}) = \frac{p_1}{(2\pi)^{\frac{n}{2}}|\Sigma_1|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(\boldsymbol{y_t}-\boldsymbol{\mu_1})^T\Sigma_1^{-1}(\boldsymbol{y_t}-\boldsymbol{\mu_1})}$$

$$d_2(\boldsymbol{y_t}) = \frac{p_2}{(2\pi)^{\frac{n}{2}}|\Sigma_2|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(\boldsymbol{y_t}-\boldsymbol{\mu_2})^T\Sigma_2^{-1}(\boldsymbol{y_t}-\boldsymbol{\mu_2})}$$

$$\vdots$$

$$d_k(\boldsymbol{y_t}) = \frac{p_k}{(2\pi)^{\frac{n}{2}}|\Sigma_k|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(\boldsymbol{y_t}-\boldsymbol{\mu_k})^T\Sigma_k^{-1}(\boldsymbol{y_t}-\boldsymbol{\mu_k})}$$

The classifier that yields the highest likelihood becomes the most likely model responsible for the current block's features, leading us to tentatively label the block as this most likely class.

Other classifiers that work especially well are Euclidean distance (in the k-means algorithm, for example) and the Mahalanobis distance, both written below:

Euclidean: $d_i(\boldsymbol{y_t}) = (\boldsymbol{y_t} - \boldsymbol{\mu_i})^2$

Mahalanobis: $d_i(\boldsymbol{y_t}) = (\boldsymbol{y_t} - \boldsymbol{\mu_k})^T\Sigma_k^{-1}(\boldsymbol{y_t} - \boldsymbol{\mu_k})$

We prefer the Gaussian classifier because it mirrors the Gaussian mixture model employed in our target distribution, $p(\boldsymbol{y_t}|x_t)$.

After we have classified all m blocks (or all m sets of features) in this manner, we recompute the statistics $\{\boldsymbol{\mu_i}, \Sigma_i, p_i\}$ to improve the decision models for each class, where

$\boldsymbol{\mu_i}$ = mean likelihood of the class i Gaussian model,
$\boldsymbol{\Sigma_i}$ = covariance matrix of the class i Gaussian model, and
p_i = proportion of blocks that fall into class i.

Repeating this process until the statistics for all k Gaussian models cease to change noticeably, we converge to a set of statistics from which we can compute the probability distributions necessary for forward recursion.

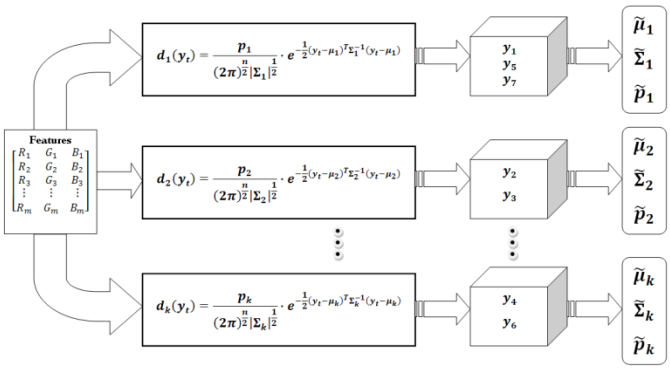

Fig. 7. Classification using Gaussian likelihoods. Statistics from the resulting clusters update the Gaussian decision function on the next iteration.

The statistics, under a Gaussian assumption, establish the probability that any feature vector $\boldsymbol{y_t}$ will arise from each class. Although continuous distributions like the Gaussian generally require integration to convert to meaningful probability, we will simply evaluate the Gaussian. Given that the current block *t* of pixels belongs to class i, the probability of observing feature vector $\boldsymbol{y_t}$ in this block is approximately

$$p(\boldsymbol{y_t}|x_t = class_i) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_i|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(\boldsymbol{y_t}-\boldsymbol{\mu_i})^T\Sigma_i^{-1}(\boldsymbol{y_t}-\boldsymbol{\mu_i})}$$

Once we have classified all of our training image features according to the Bayes-Gauss decision metric outlined above, we can empirically compute the number of transitions from each state to each other state, thereby generating an approximation to the transition probabilities:

$$p(x_{t+1}|x_t) = p\big(x_{t+1} = class_j \big| x_t = class_i\big)$$

$$p(x_{t+1}|x_t) = \frac{nBlocks: \{(x_{t+1} = class_j) \cap (x_t = class_i)\}}{nBlocks: (x_t = class_i)}$$

We then feed $p(\boldsymbol{y_t}|x_t)$ and $p(x_{t+1}|x_t)$ into the forward recursion algorithm, which we detail next.

V. The Forward Recursion Algorithm

Our goal is to find the class label x_t for all image blocks *t*. If we characterize an image by its sequence of feature vectors, the class label that should be assigned to a particular image block *t* is the class label that maximizes the *a posteriori* probability of obtaining a class label given the sequence of feature vectors observed thus far. In other words, for an image block *t*, the best estimate for its class label is:

$$\hat{x}_t = \underset{x_t}{\operatorname{argmax}}\, p(x_t|\boldsymbol{y^t})$$

Since the entire feature vector y_t is continuous-valued, computing $p(x_t|\boldsymbol{y^t})$ directly is practically infeasible. Instead, the posteriori probability is derived iteratively using forward recursion, an efficient technique to compute the distribution of X_t given $\vec{Y}_t$.

Forward recursion emerges from a reformulation of the desired probability mass function:

$$p(x_t|\boldsymbol{y^t}) = \frac{p(x_t, \boldsymbol{y^t})}{p(\boldsymbol{y^t})} = \frac{p(x_t,\ \boldsymbol{y_t}, \boldsymbol{y^{t-1}})}{p(\boldsymbol{y_t}, \boldsymbol{y^{t-1}})} = \frac{p(x_t,\ \boldsymbol{y_t}|\boldsymbol{y^{t-1}})}{p(\boldsymbol{y_t}|\boldsymbol{y^{t-1}})}$$

$$p(x_t|\boldsymbol{y^t}) = \frac{p(x_t,\ \boldsymbol{y_t}|\boldsymbol{y^{t-1}})}{p(\boldsymbol{y_t}|\boldsymbol{y^{t-1}})} = \frac{p(x_t|\boldsymbol{y^{t-1}})p(\boldsymbol{y_t}|x_t, \boldsymbol{y^{t-1}})}{\sum_{x_t'} p(x_t'|\boldsymbol{y^{t-1}})\, p(\boldsymbol{y_t}|x_t', \boldsymbol{y^{t-1}})}$$

$$p(x_t|\boldsymbol{y^t}) = \frac{p(x_t|\boldsymbol{y^{t-1}})p(\boldsymbol{y_t}|x_t, \boldsymbol{y^{t-1}})}{\sum_{x_t'} p(x_t'|\boldsymbol{y^{t-1}})\, p(\boldsymbol{y_t}|x_t', \boldsymbol{y^{t-1}})}$$

We define the desired probability distribution of classes given features as

$$\alpha_t(x_t) \overset{\text{def}}{=} p(x_t|\boldsymbol{y^t})$$

$$\beta_t(x_t) \overset{\text{def}}{=} p(x_t|\boldsymbol{y^{t-1}})$$

Forward recursion alternatingly calculates the probability mass functions $\alpha_t(x_t)$ and $\beta_{t+1}(x_{t+1})$ for block *t* in an image, where we initialize, based on probabilities accumulated during parameter estimation,

$$\beta_1(x_1) = p(x_1),$$

allowing us to iterate as follows:

$$\alpha_t(x_t) \triangleq p(x_t|\boldsymbol{y^t}) = \frac{\beta_t(x_t)p(\boldsymbol{y_t}|x_t)}{\sum_{x_t'} \beta_t(x_t')p(\boldsymbol{y_t}|x_t')}$$

$$\beta_{t+1}(x_{t+1}) \triangleq p(x_t|\boldsymbol{y^{t-1}}) = \sum_{x_t'} \alpha_t(x_t')p(x_{t+1}|x_t')$$

Parameter estimation unearths the two conditional distributions

$$p(\boldsymbol{y_t}|x_t) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma_i|^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(\boldsymbol{y_t}-\boldsymbol{\mu_i})^T\Sigma_i^{-1}(\boldsymbol{y_t}-\boldsymbol{\mu_i})}$$

$$p(x_{t+1}|x_t) = \frac{nBlocks:\ \{(x_{t+1} = class_j)\ \cap\ (x_t = class_i)\}}{nBlocks:\ (x_t = class_i)}$$

All the terms necessary for these iterative calculations are either recursive defined (in previously computed α_t or β_t values) or readily available from parameter estimation **[1]**.

VI. Using Forward Recursion for Image Block Labeling

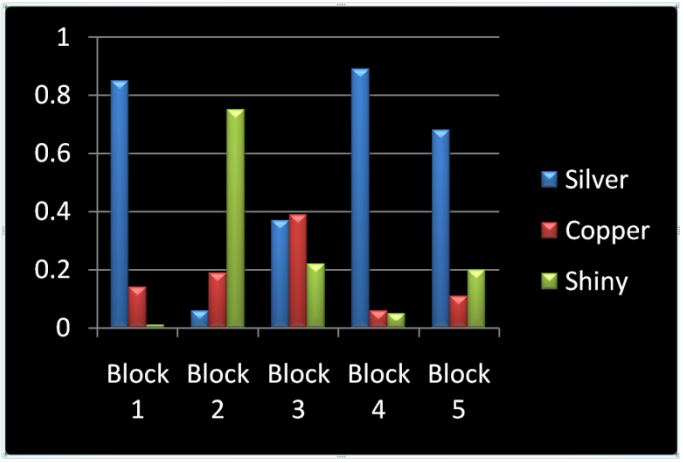

Fig. 8. Classification into one definitive label from a prob. distribution.

Forward recursion leaves us with a distribution $p(x_t|\boldsymbol{y^t})$ for each block, but we want a *single* label to describe the state of the block. Thus, the final step in classification involves vector quantization yet again; only now, unlike our work with the Bayes-Gauss likelihood functions, our classifiers have already been numerically evaluated, as shown in the chart. All that remains is automatic selection of each block's class, which our algorithm achieves by selecting the class with the highest probability in the distribution $p(x_t|\boldsymbol{y^t})$. In short, the class x_t that yields the highest probability $p(x_t|\boldsymbol{y^t})$ in block *t* is the label assigned to it.

For example, given the distributions charted in Fig. 8, this classification step would assign the first label (silver) to blocks 1, 4, and 5, the second label (copper) to block 3, and the third label (shiny) to block 2. Following this final step, all features have been successfully and optimally classified, as shown below.

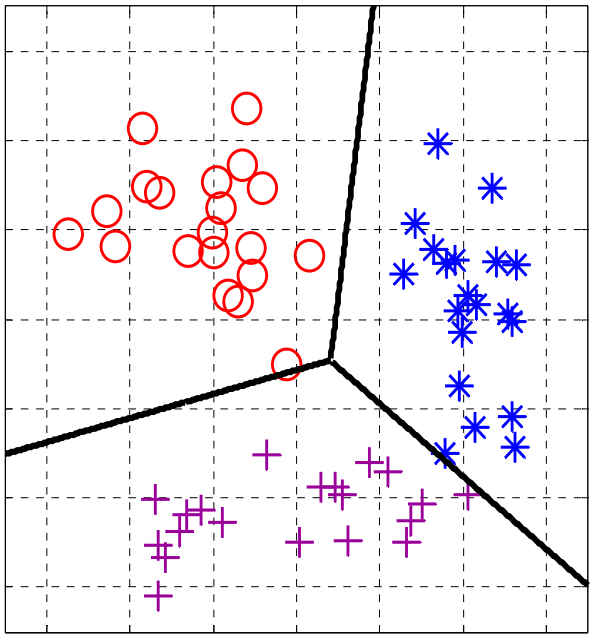

Fig. 9. Successful classification in feature space following labeling.

VII. Experimental Results and Analysis

A. Parameter Selection

This block-based classification algorithm requires a number of tuning parameters, which we select empirically:

1) The number of classification classes k, which corresponds to the number of hidden states in the HMM model, needs to be decided *a priori*. Experiments show that a classifier with three to eight states is sufficient for most of our test cases.

2) The size of the image block is a tradeoff parameter in determining the classification resolution and computational complexity. Our block sizes ranged from 2×2 to 10×10.

3) The choice of image features is an important consideration in classification. In our experiments, we considered feature descriptors such as mean color intensity values, 2D Haar wavelet coefficients, DCT coefficients, block texture or smoothness $\left(1 - \frac{1}{1+\sigma^2}\right)$ **[9]**, and response to a structuring element such as the radius-2 circle or the radius-1 diamond:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \text{ or } \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

4) The selection of training images is crucial to the performance of the classification of the test image. The training images should be similar to the test image so that similar features shared between the training and test images cluster into the same classes.

5) Image features can be accumulated by traversing the image in horizontal, vertical, or zigzag diagonal fashion. However, experiments show that the scan direction exerts minimal impact on the classification results.

B. Car & Landscape Image Classification

1) Parameter Selection:

We apply the parameter estimation algorithm to a photo taken by a camera-mounted car driving down a road **[2]**. Initially, we presume five classes in the image. The feature vector comprises average intensity of the red, green, and blue color channels.

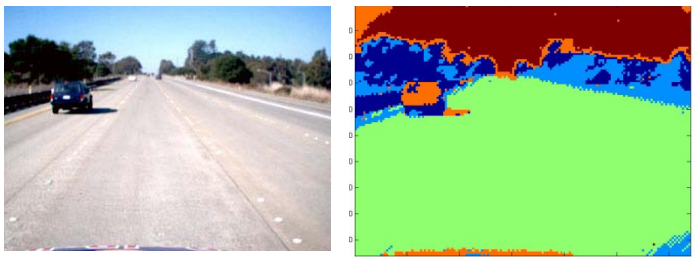

Fig. 10. Left: car training image. Classes: sky, road, car (edges), and forest.

2) Results:

As the classification result in Fig. 10 reveals, major regions in the image such as road, sky, and tree are correctly categorized as separate distinct classes. We can discriminate the rather homogeneous road and sky regions using single class labels; however, the tree and vehicle region exact multiple class labels.

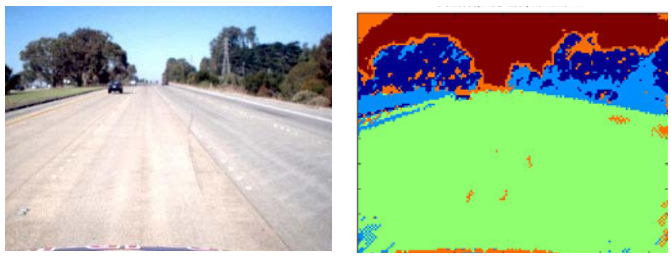

Fig. 11. Left: car test image. Classes: sky, road, car (edges), and forest.

Fig. 11 depicts the test image and its classification result. The test image is taken from the same image sequence as the training image. As expected, the classification of the test image parallels that of the training image; the road, sky, and tree regions in the test image are identified with the same class labels as the road, sky, and tree regions in the training image.

3) Extension to Two-Car Classification:

Even in the presence of two cars, test image classification continues to assign classes logically:

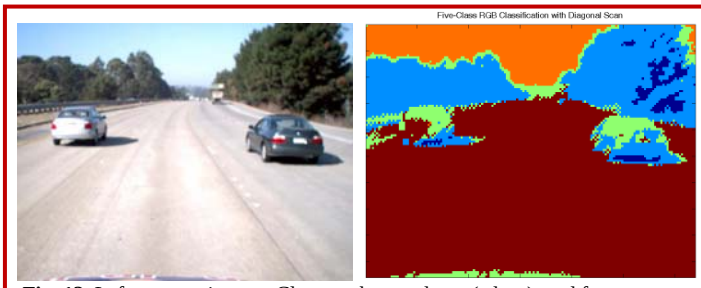

Fig. 12. Left: car test image. Classes: sky, road, car (edges), and forest.

C. Coin Image Classification

1) Parameter Selection:

Next, we tested the classifier on a set of coins images. The image classifier is trained using three photos featuring piles of coins under various lighting conditions, exhibited in Fig. 13. We prescribe a triad of classes in the training image, which we partitioned into multiple 2×2 image blocks.

2) Results:

The three classes that emerge correspond to dark-copper-colored pennies, silver-colored nickels and quarters, and high-luminosity shiny coins. In some cases, the engravings on the coins also emerge through labeling.

Fig. 13. Training images from different sources, test image boxed.

The resulting classification, based on the assignment of blocks to categories resembling copper (pennies), silver (dimes, nickels, quarters), and extremely shiny coins presents itself quite intuitively due to fine blocking structure:

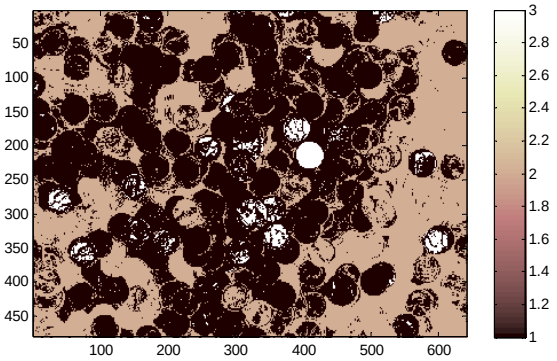

Fig. 14. Successful classification of coins into copper, silver, and super-shiny.

D. Aerial Image Classification using Color Intensities (RGB)

1) Parameter Selection:

Lastly, we applied our image classification algorithm to a series of aerial images **[3]**. The classifier is trained on one 600 × 600 color image, pictured in Fig. 15(a). We set six different classes, distinguished by their RGB image features. The 4 × 4 image blocks are traversed diagonally.

2) Results:

When the test image is very similar to the training image, classification is very consistent. Landmass is labeled as dark blue and dark red, while the water comprises four different class labels. When the test image depicts a geographical area different from the training image, classification consistency diminishes. As shown in Fig. **Fig.** 15(c), the red class, which depicts coastal regions in the training image, is the class label for water and lightly forested (pale green) regions in this test image. The other four class labels fall into desuetude. In general, this degenerate behavior insinuates a poor choice of initial conditions for parameter estimation, or an overabundance of class labels with comparatively low feature vector cardinality.

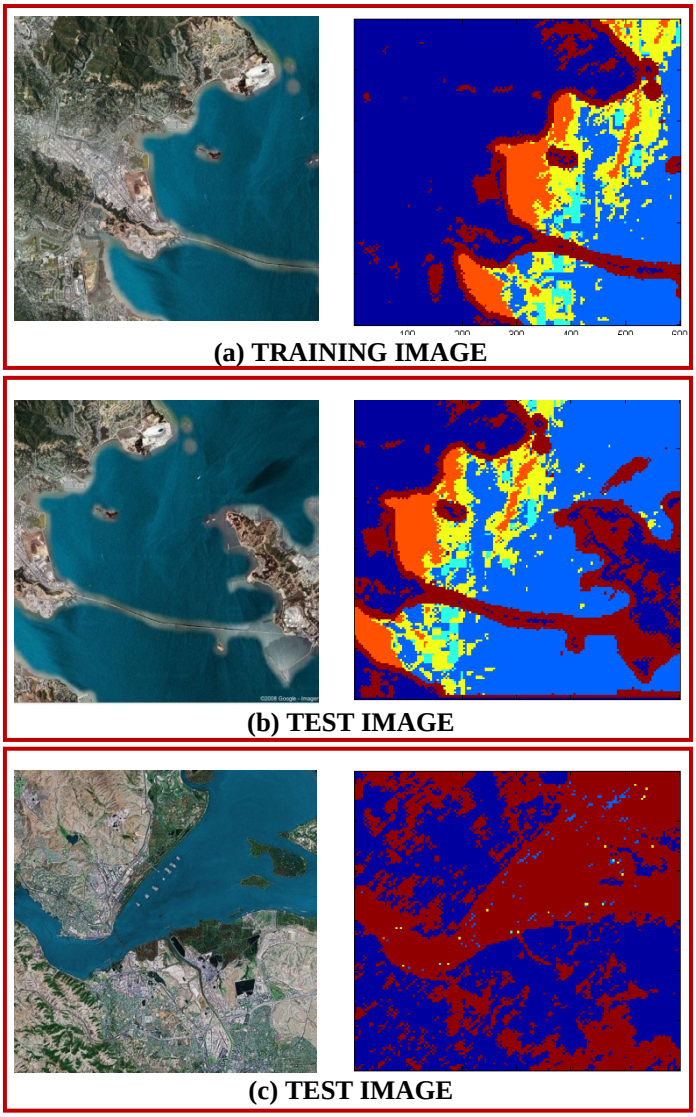

Fig. 15. Aerial images classified by block-mean RGB intensities.

E. Aerial Image Classification using DCT Coefficients

1) Parameter Selection:

As proposed by J. Li, et al, DCT coefficients serve as noteworthy features for images rich in detail and manmade structure **[4]**. Reducing the number of class labels from six to four and increasing the block size from 4 × 4 to 10 × 10 prevented overclassification from noisy local terrain or texture variations. Let the DCT coefficients be written $D_{i,j}$ for all $(i,j) \in \{1,2,\dots,9\}$. The feature vector $\boldsymbol{y_t}$ comprises $|D_{0,0}|$, $|D_{0,1}|$, $|D_{1,0}|$, and $|D_{8,8}|$, four distinct DCT coefficients.

2) Results:

The classification results remain more consistent than the previous RGB classifier. Using DCT coefficients, the classifier better resists slight color variations in the water region, as the images on the following page corroborate:

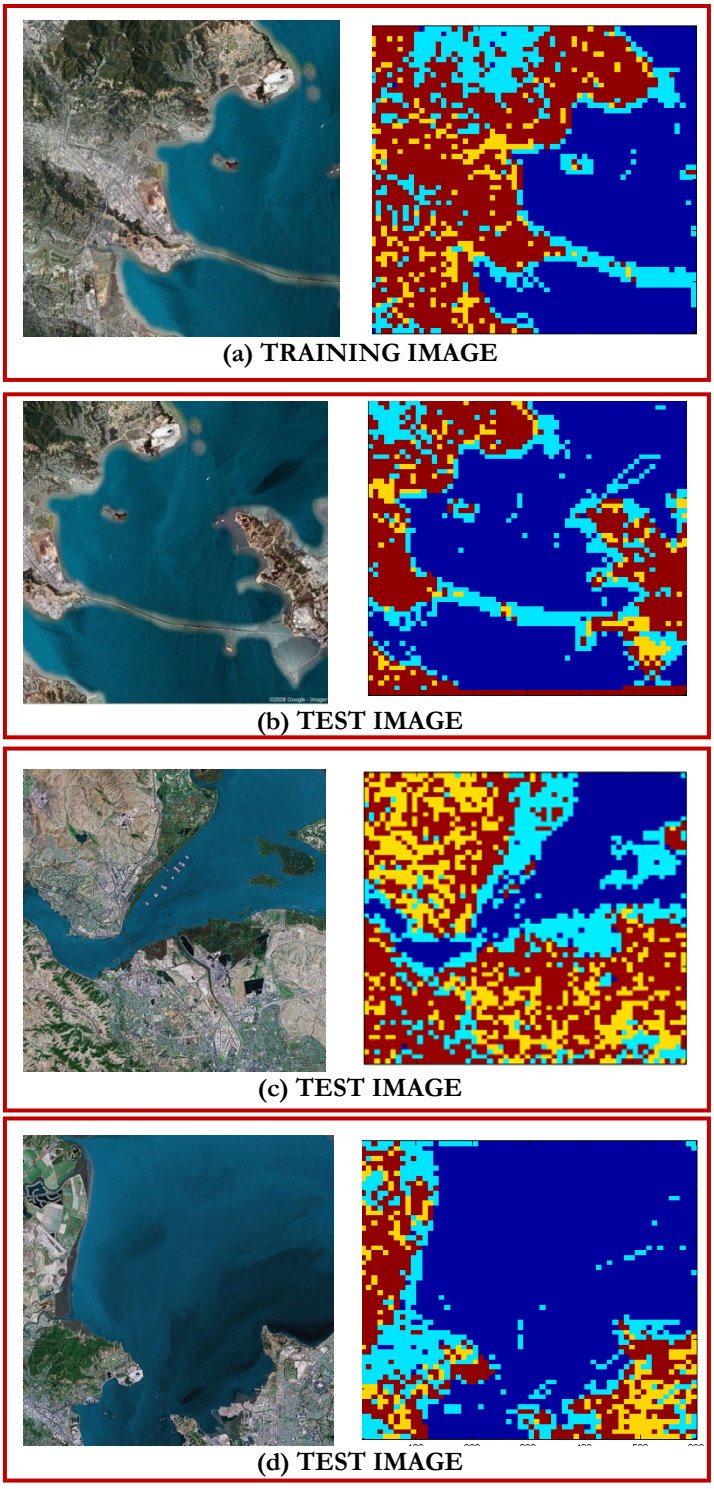

Fig. 16. Aerial images classified using DCT coefficient quartet.

VIII. Conclusion

The image classification algorithm described in this paper models the probability distribution of observed feature vectors given the class as a Gaussian density function, with mean and covariance matrix estimable through iterated Lloyd-Max vector quantization. Gaussian decision functions mirror our Gaussian observation density. Classification itself relies on maximizing the likelihood of the state distribution generated from forward recursion using statistics acquired in iterative Bayesian parameter estimation.

Feature selection is integral to classification, as one type of feature might suit one set of images but fail to fit another. While classification is not particularly time intensive, it would be interesting to juxtapose the classification performance and computational efficiency of forward recursion and the Viterbi algorithm.

The applications illustrated through our limited sample of experimental results only touch the surface of all the uses that classification boasts in science and engineering. Car-mounted cameras can supply an autonomous driver with invaluable information about all sorts of obstacles, thereby preventing collision with anything suggestive of solidity, such as lightposts, medians, rocks, and other cars. Classifying large piles and collections of superficially similar objects such as coins and jewelry might lead to counterfeit exposal or higher order. Aerial photographs beg for military reconnaissance or even use in the study of extraterrestrial surfaces with radar imaging.

Applying these techniques to photographs of nature can further empower the scientific observer with a novel but nevertheless systematic taxonomy, classifying species of fauna as well as a wide variety of flora, all possible through camera software. Indeed, unsupervised classification's greatest promise to the scientific community is its ability to detect underlying structure and bring it to the surface with minimal human influence. Whereas variables remain – the number of classes we must prescribe, the features we anticipate separating our categories, and the precision of our observation blocks – the algorithm remains largely automated and order-driven.

In an age of oft-careless experimental design, misplaced human anticipation, and hasty conclusion, the ability to detect order with minimal human input is appealing not only for the insight it can provide to otherwise indecipherable scenes but also its independence from and immunity to our preconceived notions. Our exploration represents only the beginning, as humankind has just begun to tap the power that is natural classification. The world is literally a hidden Markov model, and forward recursion can help us uncover its dissembled mysteries.

References

[1] K. Ozonat. "Nonlinear and Non-WSS Estimation." Lecture notes, Wednesday, April 30, 2008.

[2] S. Thrun. Road data from Computer Vision homework assignment #1, http://robots.stanford.edu/cs223b06/homework/hw1/SOURCE/.

[3] Google Maps satellite view, downloaded May 23, 2008.

[4] J. Li, A. Najmi, and R.M. Gray. "Image Classification by a Two-Dimensional Hidden Markov Model." *IEEE Transactions on Signal Processing*, vol. 48 no. 2, pp. 517-533, February 2000.

[5] K. Ozonat and R.M. Gray. "Vector Quantization for Image Classification with Side Information for the Additive Gaussian Noise Channels." Stanford University EE Dept. ICIP (3) 2005.

[6] K. Ozonat and R.M. Gray. "Fast Gauss Mixture Image Classification Based on the Central Limit Theorem." Stanford University EE Dept. IEEE 6th Workshop on Multimedia Signal Processing, 2004.

[7] B. Porat. *Digital Processing of Random Signals*. Mineola, New York: Dover Publications, 2008.

[8] R.C. Gonzalez, and R.E. Woods. *Digital Image Processing, Third Edition*. Upper Saddle River, NJ: Prentice Hall, 2007.

[9] R.C. Gonzalez, R.E. Woods, and S.L. Eddins. *Digital Image Processing using Matlab*. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[10] B. Girod. Lecture Notes, EE 368: Digital Image Processing, spring 2007.

Acknowledgments

We would like to thank Kivanc Ozonat and Rajiv Agarwal for sharing their knowledge, literature, and expertise with image classification and forward recursion. We appreciate their patience with our questions and their contagious passion in statistical signal processing. This document would not be possible without their continued guidance and invaluable input.