

*Title:*

**AFREET: Human-Inspired Spatio-Spectral Feature Construction for Image Classification with Support Vector Machines**

*Author(s):*

Simon Perkins and Neal Harvey

*Submitted to:*

<http://lib-www.lanl.gov/la-pubs/00357127.pdf>

---

# AFREET: Human-Inspired Spatio-Spectral Feature Construction for Image Classification with Support Vector Machines

---

**Simon Perkins**

S.PERKINS@LANL.GOV

Space and Remote Sensing Sciences, Los Alamos National Laboratory, NM 87545 USA

**Neal Harvey**

HARVE@LANL.GOV

Space and Remote Sensing Sciences, Los Alamos National Laboratory, NM 87545 USA

## Abstract

We examine the task of pixel-by-pixel classification of the multispectral and grayscale images typically found in remote-sensing and medical applications. Simple machine learning techniques have long been applied to remote-sensed image classification, but almost always using purely spectral information about each pixel. Humans can often outperform these systems, and make extensive use of spatial context to make classification decisions. We present AFREET: an SVM-based learning system which attempts to automatically construct and refine spatio-spectral features in a somewhat human-inspired fashion. Comparisons with traditionally used machine learning techniques show that AFREET achieves significantly higher performance. The use of spatial context is particularly useful for medical imagery, where multispectral images are still rare.

## 1. Introduction

### 1.1 Machine Learning and Flat Image Classification

Earth-observing satellites produce vast quantities of image data every day, much of it multispectral. These images are used for a wide variety of applications, ranging from weather prediction, through agricultural use monitoring, to making maps of remote areas. One of the core tasks in much of this analysis is the identification in the image of relevant objects of interest: clouds, wheat fields, roads, and so on.

Medical imagery, from sources such as X-rays, microscope slides, CAT scans and MRI scans, shares many

characteristics with remote-sensed imagery. Again, vast quantities of it are generated and must be analyzed to find objects of medical interest: tumors, unhealthy cells, particular kinds of tissue, etc.

Both these types of imagery can be called “flat”. They have an essentially 2-D nature, and are usually imaged at a fixed (and known) scale. Many of the classification tasks in these domains can be reduced to the problem of performing an initial pixel-by-pixel classification of a given image, which is then used for further analysis.

Given the vast quantity of image data generated in these two fields, it is clear that reliable automated techniques for pixel-by-pixel classification would be of enormous benefit. At present, most medical imagery is classified by human experts,<sup>1</sup> but in the remote-sensing domain, automated classifiers are widely used. Most of these classifiers are designed by hand, a slow and laborious process which requires detailed and accurate knowledge of the physics of the object being sought, and of the sensor used to produce the image, and also of the background against which the object will be imaged. If the sensor is upgraded, or its calibration drifts, or if the task requirements are modified even slightly, a redesign of the classifier is usually necessary.

Given the difficulty of hand-designing classifiers, it is natural to look at machine learning and pattern recognition techniques, and ask if they might allow us to generate reliable automatic classifiers much more quickly and easily. Many researchers have examined these approaches, ranging from simple statistical methods such as minimum distance and maximum likelihood classifiers (Richards, 1993), to more complex approaches such as neural networks (Bischof & Leonardis, 1998) and, more recently, Support Vector Machines (Roli & Fumera, 2000). While high perfor-

---

<sup>1</sup>And most patients prefer it that way...

mance is often achieved with these systems, it is still often or usually the case that a human “eyeballing” the image, visualized in appropriate false colors where appropriate, can do better.

## 1.2 How Do Humans Do It?

One problem with most learning systems that have been developed for pixel-by-pixel image classification is that they base their decisions purely on the spectral information contained in each pixel. Humans, in contrast, have relatively little ability to perceive spectral information, being limited to at most three channels. Instead, they appear to make great use of spatial context and texture information in making decisions. This effect is even more extreme in medical imagery, which is usually monospectral, and where the raw intensity value in each pixel carries very little information about its identity.

Clearly, if we are going to use machine learning to produce classifiers that can compete with humans, then we need to find a way of incorporating spatial context information into the classifier. One simple way is to provide extra channels for each pixel, each describing some aspect of the local neighborhood. For instance we could apply a simple smoothing mask to each spectral channel of the image, and then incorporate the smoothed channels as extra elements of the classifier feature vector. Gong and Howarth (1990) describes an example where an extra channel of texture information was used to improve performance. However, pre-defined spatial context features can only represent a limited amount of context information. We can attempt to include many different texture and statistical context features for many different sized neighborhoods, but we rapidly run into problems. Simply generating all those extra features from an image can take enormous amounts of time, and many of the features may be redundant or carry no useful information.

One way of tackling the problem is to look at what human experts do when asked to design an image classification algorithm by hand. Typically they isolate useful information from the images by applying sequences of standard image processing operations, including smoothing masks, morphological operations and texture operators, and then make a decision using relatively simple thresholds and logic. This process suggests a framework for generating an almost infinite number of possible spatio-spectral features for an image, by chaining together a set of standard image processing operations. These features can then be passed on to a relatively unsophisticated learning system for final classification.

The trick, of course, is to decide which spatio-spectral features to generate. In related work, Draper et al. (1999) treat constructing good features as a control problem. They use a reinforcement learning approach to pick primitive operations to chain together. We present a different approach to the same basic problem, combining ideas from Support Vector Machines and Evolutionary Computation.

## 2. AFREET

### 2.1 Motivations

Our current work on AFREET is motivated by the ideas presented above, and by earlier work on a Genetic Programming system for image classification, called GENIE.<sup>2</sup> For complete details, see, e.g. (Perkins et al., 2000; Theiler et al., 1999), but suffice to say that GENIE uses a fairly standard Genetic Algorithm (Holland, 1975) to evolve a population of image classifiers. Each classifier performs a sequence of primitive image processing steps that transforms the raw image data planes into a set of “answer planes”. A linear discriminant, derived by finding the Fisher Discriminant (Bishop, 1995) on the training data, is then used to generate a final binary classification for each pixel.

Despite being a relatively unsophisticated algorithm, GENIE produces extremely good classifiers (see, e.g. (Harvey et al., 2000)). Its success seems to stem from the rich variety of features it is able to construct from the primitive genes, which allows the Fisher Discriminant “backend” to do a good job of classification. However, GENIE makes no explicit attempt to produce classifiers that generalize well, and training times can be very long (many hours). AFREET was developed primarily to address these two problems.

### 2.2 Design Details

#### 2.2.1 CLASSIFIER STRUCTURE

Our earlier program, GENIE, works with a population of classifiers. As training progresses, the population tends to converge towards a single solution and the population will contain many functionally identical chromosomes. Evaluating all of these chromosomes involves a lot of redundant effort.<sup>3</sup> In contrast, AFREET works with a single classifier and attempts to iteratively refine it. This classifier consists of a bank of “feature generators” which transform the raw input

---

<sup>2</sup>See (Banzhaf et al., 1998) for an good introduction to GP. Note that GENIE is a relatively non-traditional GP system.

<sup>3</sup>Note however that *completely* identical chromosomes are not re-evaluated.

image planes into a set of “feature planes”. A linear discriminant is then applied on a pixel-by-pixel basis to the feature planes to produce a final binary classification plane. The general structure of this classifier is shown in Figure 1.

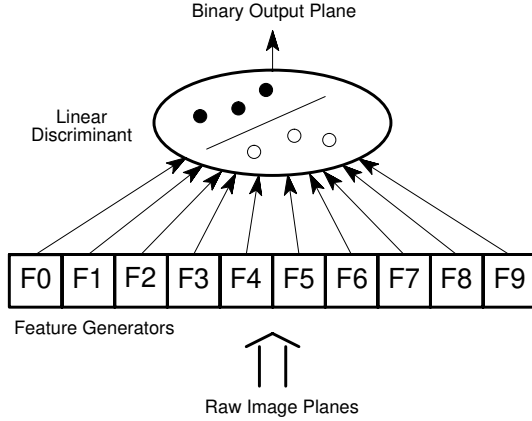


Figure 1. Structure of the classifiers developed by AFREET. The raw image planes are used to derive a number of feature planes, labeled F0 to F9 which are combined using a linear discriminant to give a final binary classification.

Each feature generator is represented as a program tree, a representation commonly used in genetic programming systems. The primitive operations that are available in Afreet are listed in Table 1. All of the operators take zero or more images as input, and produce a single image as output. Figure 2 shows a typical tree that might generate one of the feature planes in Figure 1.

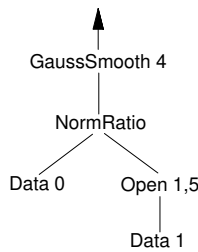


Figure 2. A typical feature generator used to generate one feature plane. This particular generator performs a morphological opening on plane 1 of the input image, using a linear structuring element of radius 5. It then finds the normalized ratio between this result and input image plane 0. Finally this ratio image is smoothed with a Gaussian mask of radius 4.

Before the feature planes are generated, the pixel values in the input image are rescaled so that the minimum value is 0.0 and the maximum value is 1.0. All

operators used in AFREET assume that input pixel values are of the order of unity and  $\geq 0$ , and they produce output which has the same properties.<sup>4</sup> Once a feature plane is generated it is then rescaled again to have a mean value of 0.0 and a standard deviation of 1.0. The various normalization scales and offsets are only calculated during training. When the trained classifier is being applied to new images, the previously derived normalization values are re-used.

### 2.2.2 INITIALIZATION

The set of feature generators is generated randomly at the start of training subject to two constraints: (a) the feature set contains no duplicate generators, and (b) the depth of the generators does not exceed a user-defined limit, typically taken to be 3. The smallest possible tree has a depth of 1. We use a generation technique that encourages compact trees, by linearly increasing the probability of a terminal node being selected towards 1.0, as the depth limit is approached.

### 2.2.3 TRAINING THE DISCRIMINANT

The linear discriminant is trained as a Support Vector Machine (Vapnik, 1995; Burges, 1998), using a modified version of Platt’s SMO algorithm (Platt, 1999), suggested by Keerthi et al. (1999). Although this algorithm allows us to train kernel SVMs as well as linear ones, we have chosen to use only linear discriminants, because it makes feature set evolution much easier, as explained below.

Our training data consists of training images, in which pixels have been marked as belonging to a “positive” class ( $y_i = +1$ ), or a “negative” class ( $y_i = -1$ ). In general there will be different numbers of pixels in each class. We have found in our applications that better results are often achieved if we try equally hard to get both categories correct, rather than simply minimizing total misclassifications. Therefore we use a cost function that makes the total importance of the “positive” pixels equal to the total importance of the “negative” pixels. This is easily done by modifying the usual SVM objective function to be optimized to the following:

$$\frac{1}{2} \|\mathbf{w}\|^2 + \sum_i C_i \xi_i \quad \begin{cases} C_i = \frac{K}{n_+} & \text{if } y_i = +1 \\ C_i = \frac{K}{n_-} & \text{if } y_i = -1 \end{cases} \quad (1)$$

where  $\xi_i$  are the “slack variables”,  $\mathbf{w}$  is the vector of weights describing the linear discriminant,  $K$  is a constant,  $y_i$  is the class label associated with the  $i$ th pixel, and  $n_+$  and  $n_-$  are the numbers of pixels in the positive and negative classes respectively. In the more

<sup>4</sup>This accounts for the slightly strange formula for the NormRatio operator.

Table 1. Primitive operations used to construct AFREET features. **Data** is the basic operation used to access the input image. All other operations are neighborhood operations, except **Peak** and **NormRatio**, which work on single pixels. **Open** and **Close** may use a linear structuring element. The relevant morphological operation is carried out with this S.E. at all possible distinct orientations on the discrete pixel grid, and the output value is the maximum of all possible openings, or the minimum of all possible closings, as appropriate. Consult any good image processing textbook for more details. In the descriptions below, the **radius** parameter can take values between 1 and 10; the **center** parameter takes a value between 0.0 and 1.0; and the **shape** parameter can takes values of **DISK** or **LINE**.

NAME	INPUTS	PARAMS	DESCRIPTION
<b>Data</b>	0	<b>index</b>	Extracts raw data plane <b>index</b> from the input image.
<b>GaussSmooth</b>	1	<b>radius</b>	Gaussian smoothing with a kernel of the specified <b>radius</b> .
<b>Grad</b>	1	<b>radius</b>	Smoothed gradient magnitude using Gaussian smoothing with a kernel of the specified <b>radius</b> .
<b>Min</b>	1	<b>radius</b>	Minimum value within <b>radius</b> pixels of each pixel
<b>Max</b>	1	<b>radius</b>	Maximum value within <b>radius</b> pixels of each pixel
<b>StdDev</b>	1	<b>radius</b>	Standard deviation in circular neighborhood of given <b>radius</b> .
<b>Peak</b>	1	<b>center</b>	Non-linear transfer function. Pixel values are mapped onto new values given by a Gaussian function with the given <b>center</b> and standard deviation 0.25.
<b>Open</b>	1	<b>shape, radius</b>	Morphological opening. <b>shape</b> determines if a disk or linear structuring element is used. <b>radius</b> determines the size of the S.E.
<b>Close</b>	1	<b>shape, radius</b>	Morphological closing. Parameters as for <b>Open</b> .
<b>NormRatio</b>	2	—	Normalized ratio between two planes: $(\frac{a-b}{a+b} + 1) \times 0.5$

usual SVM formulation, a single constant  $C$  is used for all pixels. See (Burges, 1998) for more details.

#### 2.2.4 EVOLVING THE FEATURE SET

It is unlikely that the randomly generated initial feature set constitutes an ideal basis for classification, so AFREET attempts to iteratively refine the features using the algorithm shown in Figure 3. The algorithm essentially involves deciding heuristically which features are the most important and least important, and then either replacing the least important with randomly generated new features, or replacing the most important with small “mutations” of that feature. If the change produces a significant decrease in the value of the SVM objective function, then it is kept, otherwise we revert back to the unmodified feature. If the objective function is essentially unchanged (to within 1%), then the new feature is kept if it is computationally less expensive than the old one. This refinement strategy is essentially a greedy one. For the feature refinement process, we save time by only training on a randomly chosen subset of all the training points, typically of size 10,000. Once a feature set has been chosen, all the points are used for the final optimization.

The importance of a feature is decided simply by looking at the magnitude of the weight vector component associated with that feature. Since all features are normalized to have the same mean and standard deviation, this component provides some indication of how

important a particular feature is. Note that we only have direct access to the weight vector for linear SVMs. Linear SVMs are also preferred because training times are much faster using SMO. We use an evolutionary computing method called “tournament selection” to select the “best” and “worst” feature planes, which introduces some stochasticity into the selection process. The tournament size is based on the current feature set size, and is chosen to give a fixed probability (typically 0.25) of choosing the absolute best or worst feature.

Three kinds of mutations are employed with equal probability: (a) **Parameter Mutation**: A single node with a parameter is picked, and that parameter is mutated in a manner appropriate to the parameter type. (b) **Grow Mutation**: A new node is generated randomly and inserted at the head of the generator tree. The old tree becomes one of its arguments. If the new node has more than one argument, the other arguments are generated randomly. (c) **Shrink Mutation**: The opposite of Grow: The head node is removed, and one of its arguments is selected randomly to be the new feature tree.

These mutations are partially inspired by the ways in which a human would typically modify an image processing pipeline, while experimenting with possibilities, but are obviously less well directed. Mutations that would cause the tree depth to exceed a user-defined depth limit (greater than the initialization depth limit) are not allowed. This absolute depth

```

Randomly initialize features
Perform initial optimization
for  $j = 1$  to  $F$ :
  let  $p_m = j/F$ 
  let  $T = \ln \frac{1-p_a}{2} / \ln \frac{n-1}{n}$ 
  Randomly choose  $T$  feature indices  $I_1 \dots I_T$ ,
                                     with replacement

  with probability  $p_m$ :
    Find  $i$  such that  $|w_{I_i}|$  is maximized
    Mutate feature with index  $I_i$ 
  else:
    Find  $i$  such that  $|w_{I_i}|$  is minimized
    Randomize feature with index  $I_i$ 
  Re-optimize
  If objective function decreased, then keep change,
                                     otherwise revert

end for

```

Figure 3. The feature set refinement algorithm.  $F$  is the desired number of refinement cycles;  $p_m$  is the probability of mutating a good feature, rather than randomizing a bad one;  $T$  is the tournament size;  $n$  is the size of the current feature set;  $p_a$  is the desired probability of picking the absolute best or worst feature in the tournament;  $w_k$  is the component of the weight vector associated with the  $k$ th feature.

limit is typically set to 5.

The probability of a mutation is adjusted linearly from 0 at the beginning of the refinement process to 1 at the end. The motivation for this is that at the beginning of the run it is more important to get rid of useless features and try random replacements, while towards the end, we hopefully have quite a good set of features and so should focus on improving them.

### 2.2.5 PRUNING

One of the potential advantages of a population-based approach such as GENIE is that it begins by considering many different solutions, and then as the training run progresses, it tends to converge attention on a much smaller number of solutions. We can obtain some of this effect in AFREET by starting with an initial feature set that is much larger than the feature set we want to end up with. In conjunction with the refinement process, we perform a series of pruning steps, each of which simply eliminates the feature with the current lowest weight component and re-optimizes. Typically when using pruning, we start with a feature set ten times larger than the final feature set, and prune down to the final set size during the initial 25% of the feature refinement process.

### 2.2.6 INTERFACE ISSUES

Obtaining accurate training data is an essential part of a learning system. We use a Java GUI called ALADDIN

to allow human experts to provide training data. ALADDIN allows the user to visualize a multispectral image, and “paint” regions of positive and negative class value onto the image. Not all pixels need to be classified. The same interface is used to visualize classification results and can be used to refine the training data before re-training.

## 3. Experiments

To test and illustrate the effectiveness of AFREET we include some examples of the system being applied to problems in the remote sensing and medical domains.

### 3.1 Data Sets

Figure 4 shows the training and test data that were used. Three tasks were selected:

1. Finding major roads and parking lots in 2m resolution IKONOS satellite imagery of Los Alamos, New Mexico. The IKONOS data consists of four channels: red, green, blue and near-infrared.<sup>5</sup>
2. Finding brain tissue in MRI scans of a human head. The MRI scans are single-channel grayscale images.
3. Finding eyeball tissue in the same MRI images.

These tasks were deliberately chosen to be very difficult to do spectrally. The urban images contain many roof tops and driveways that are spectrally identical to the roads that we want to find. The single-channel MRI scans contain many other objects with the same grayscale intensity as the objects we wish to find.

For each task, three scenes were selected and labeled. We used the first two scenes (A and B) for training, and tested the resulting classifier on the third scene (C).

### 3.2 Experimental Details

Six experiments were performed for each task, to compare AFREET with purely spectral classifiers, and to test various aspects of AFREET’s operation:

<sup>5</sup>IKONOS is a commercial high-resolution imagery satellite operated by Space Imaging, Inc ([www.spaceimaging.com](http://www.spaceimaging.com)). IKONOS color images actually have a pixel resolution of 4m, but this has been “sharpened” to 1m resolution using 1m IKONOS panchromatic imagery, and then resampled to produce these 2m resolution images.

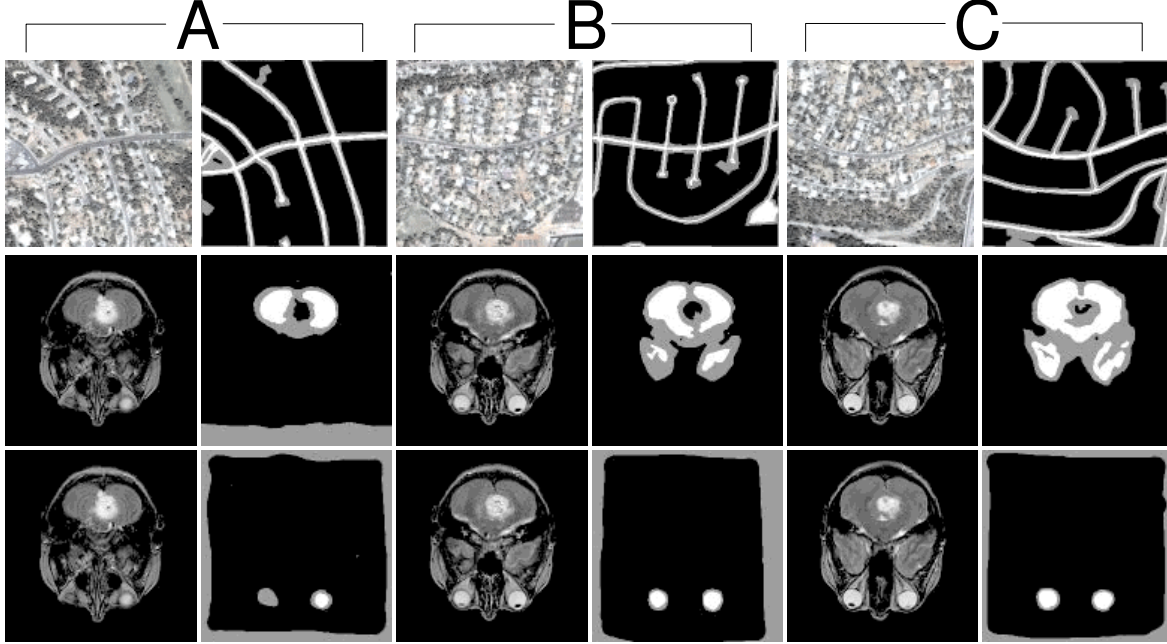


Figure 4. Training data. Each row shows the training/test images used for a different task. From top to bottom: roads, brain tissue, and eyeball tissue. To the right of each scene is shown the pixel labeling associated with that scene. White corresponds to the positive class, black corresponds to the negative, and gray means that pixel was unlabeled. The classifiers were trained on scenes **A** and **B**, and were tested on scene **C**. Note that the two medical imagery tasks use the same scenes, but with different pixel labelings.

1. Conventional spectral-only maximum likelihood classification using Normal distributions.
2. Linear Support Vector Machine trained on just the spectral channels, using the SMO algorithm.
3. AFREET, without pruning, using a feature set of size 10 and 100 feature refinement steps.
4. AFREET, with pruning, starting with a feature set of size 100 and pruning down to 10 features over the initial 25% of a 100 step feature refinement process.
5. AFREET, with no feature refinement process, but starting with a feature set of size 100, and pruning down to 10 features, i.e. simply selecting the best 10 of the initial 100 features.
6. AFREET, with no refinement process, and no pruning, i.e. simply optimizing with 10 randomly generated features.

The last two experiments are controls to test the usefulness of the pruning process by itself, and to see how useful the feature refinement process really is.

All the SVM experiments, including AFREET, used  $K = 1000$  in the objective function (expression 1). Other parameters took the “typical” values mentioned in the AFREET description above. All experiments that involved a random component (3, 4, 5 and 6) were carried out 5 times and the results averaged.

## 4. Results

Table 2 shows the experimental results. The “miss rate” gives the percentage of pixels that are really in the positive class that were misclassified. The “false alarm rate” gives the percentage of pixels in the negative class that were misclassified. The “average” gives the mean of these two numbers. Due to the weighting scheme described in 2.2.3, it is this average that is optimized during AFREET training. Both training and test scores are shown.

The general pattern of these results shows clearly that AFREET (Experiments 3 and 4) outperforms both the purely spectral linear SVM (Experiment 2), and the Maximum Likelihood classifier (Experiment 1), on both training scores and test scores. This is particularly noticeable on the medical data where the AFREET runs get average misclassification errors that are less

Table 2. Training and test scores for each of the six experiments and each of the three tasks. The table shows the miss percentage, false alarm percentage and the average of those two percentages. The uncertainties shown are the standard error of the mean of five trials of each experiment and task.

TRAINING SCORES									
EXP	ROADS			BRAIN			EYES		
	Miss %	FA %	AVG %	Miss %	FA %	AVG %	Miss %	FA %	AVG %
1	7.1	8.1	7.6	4.4	9.2	6.8	8.3	9.2	8.7
2	6.4	11.0	8.7	0.0	17.4	8.7	7.8	10.5	9.2
3	$3.5 \pm 0.4$	$3.9 \pm 0.8$	$3.7 \pm 0.6$	$0.23 \pm 0.06$	$0.41 \pm 0.07$	$0.32 \pm 0.05$	$0.0 \pm 0.0$	$0.31 \pm 0.1$	$0.16 \pm 0.05$
4	$3.8 \pm 0.5$	$3.6 \pm 0.6$	$3.7 \pm 0.4$	$0.080 \pm 0.04$	$0.23 \pm 0.05$	$0.15 \pm 0.03$	$0.0 \pm 0.0$	$0.11 \pm 0.02$	$0.06 \pm 0.01$
5	$4.2 \pm 0.2$	$4.0 \pm 0.7$	$4.1 \pm 0.4$	$0.18 \pm 0.06$	$0.30 \pm 0.08$	$0.24 \pm 0.07$	$0.020 \pm 0.02$	$0.89 \pm 0.3$	$0.46 \pm 0.2$
6	$8.2 \pm 0.6$	$9.2 \pm 0.5$	$8.7 \pm 0.4$	$1.5 \pm 0.3$	$3.6 \pm 0.8$	$2.5 \pm 0.5$	$1.1 \pm 0.5$	$7.3 \pm 1.1$	$4.2 \pm 0.8$

TEST SCORES									
EXP	ROADS			BRAIN			EYES		
	Miss %	FA %	AVG %	Miss %	FA %	AVG %	Miss %	FA %	AVG %
1	15.4	7.6	11.5	1.6	6.2	3.9	3.9	10.3	7.1
2	17.3	10.5	13.9	0.0	13.6	6.8	3.8	12.4	8.1
3	$18.1 \pm 1.7$	$4.2 \pm 0.9$	$11.2 \pm 1.3$	$0.012 \pm 0.01$	$0.040 \pm 0.02$	$0.026 \pm 0.01$	$0.40 \pm 0.4$	$0.39 \pm 0.05$	$0.40 \pm 0.2$
4	$18.0 \pm 0.9$	$4.3 \pm 0.6$	$11.2 \pm 0.4$	$0.002 \pm 0.002$	$0.24 \pm 0.13$	$0.12 \pm 0.07$	$0.11 \pm 0.11$	$0.38 \pm 0.1$	$0.24 \pm 0.07$
5	$20.4 \pm 0.9$	$5.0 \pm 1.1$	$12.7 \pm 0.9$	$0.050 \pm 0.05$	$0.26 \pm 0.2$	$0.16 \pm 0.07$	$0.020 \pm 0.02$	$0.89 \pm 0.3$	$0.46 \pm 0.15$
6	$21.5 \pm 2.4$	$10.2 \pm 0.8$	$15.9 \pm 1.1$	$1.9 \pm 0.7$	$2.2 \pm 0.8$	$2.0 \pm 0.6$	$0.26 \pm 0.2$	$9.4 \pm 1.7$	$4.8 \pm 1.0$

than 10% of the purely spectral runs. This is perhaps not that surprising since the single-channel images contain very little spectral information, and so texture is a very important cue. The one anomaly is in road finding where the Maximum Likelihood classifier achieves essentially the same performance as AFREET on the test scene, although AFREET achieves a significantly higher training score.<sup>6</sup>

Amongst the various AFREET variants, it is difficult to see a clear winner. AFREET with no feature refinement at all (Experiment 6) does consistently worse, but the other variants are not statistically different from one another. More work needs to be done to see which of “pruning from a large feature set”, or “gradual refinement of a small feature set”, is a more powerful technique.

Figure 5 compares the best classifications produced by AFREET, with the results from Maximum Likelihood classification. This shows clearly the power of using spatial context information in order to classify spectrally ambiguous image features.

<sup>6</sup>*Subjectively*, however, the AFREET test results look better, but this is not captured by the fitness metric.

## 5. Conclusions and Further Work

For many real-world image classification problems, purely spectral feature vectors are not sufficient. We have presented AFREET, an SVM-based system that attempts to automatically construct spatio-spectral feature vectors by putting together image processing operators to form pipelines similar to those hand-designed by humans. Ideas from evolutionary computing are used to refine the feature set from a random initial selection. In the experiments described here, AFREET performs a good job of classification, and clearly uses spatial context to good advantage to outperform purely spectral classifiers in most cases.

One important thing that AFREET lacks is a sense of direction when trying to mutate and replace feature generators. Humans do not make totally random changes when optimizing image processing pipelines — they make purposeful changes. We are currently examining a boosting technique that will allow AFREET to rapidly estimate the effect of adding in a different feature generator without having to re-optimize. This will allow AFREET to screen a much larger set of potential features in a shorter time, and to make refinements that are more sensible.



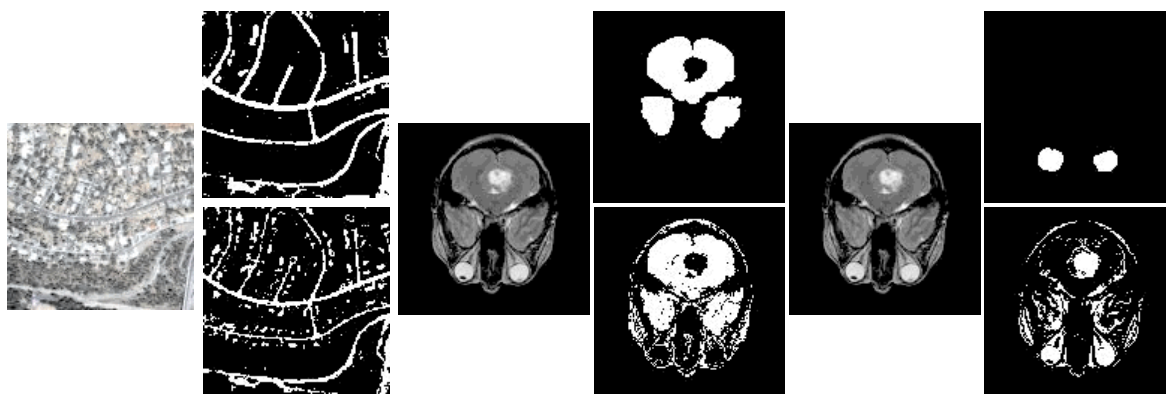


Figure 5. Result images from Experiment 4 for each task. The classification output from AFREET (top) and maximum likelihood (bottom) are shown to the right of each scene. The tasks from left to right are: roads, brain tissue, and eyeball tissue.

## References

- Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). *Genetic programming: An introduction*. San Francisco, CA: Morgan Kaufmann.
- Bischof, H., & Leonardis, A. (1998). Finding optimal neural networks for land use classification. *IEEE Transactions on Geoscience and Remote Sensing*, 36, 337–341.
- Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford University Press.
- Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2.
- Draper, B., Bins, J., & Baek, K. (1999). ADORE: Adaptive object recognition. *Proc. International Conference on Vision Systems* (pp. 522–537). Las Palmas de Gran Canaria, Spain.
- Gong, P., & Howarth, P. (1990). The use of structural information for improving land-cover classification accuracies at the rural-urban fringe. *Photogrammetric Engineering and Remote Sensing*, 56, 67–73.
- Harvey, N., Perkins, S., Brumby, S., Theiler, J., Porter, R., Young, A., Varghese, A., Szymanski, J., & Bloch, J. (2000). Finding golf courses: The ultra high tech approach. In et S. C. al. (Ed.), *Real world applications of evolutionary computing*, vol. 1803 of *Lecture Notes in Computer Science*. Springer.
- Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.
- Keerthi, S., Shevade, S., Bhattacharyya, C., & Murphy, K. (1999). *Improvements to platt's smo algorithm for svm classifier design* (Technical Report CD-99-14). Dept. of CSA, IISc, Bangalore, India.
- Perkins, S., Theiler, J., Brumby, S., Harvey, N., Porter, R., Szymanski, J., & Bloch, J. (2000). GENIE: A hybrid genetic algorithm for feature classification in multi-spectral images. In *Proc. SPIE 4120: Applications and Science of Neural Networks, Fuzzy Systems and Evolutionary Computation III*.
- Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods — support vector learning*, 185–208. MIT Press.
- Richards, J. (1993). *Remote sensing digital image analysis*. Springer-Verlag.
- Roli, F., & Fumera, G. (2000). Support vector machines for remote-sensing image classification. *Proc. EOS/SPIE Symposium*. Barcelona.
- Theiler, J., Harvey, N., Brumby, S., Szymanski, J., Alferink, S., S.Perkins, Porter, R., & Bloch, J. (1999). Evolving retrieval algorithms with a genetic programming scheme. *Proc. SPIE 3753* (pp. 416–425).
- Vapnik, V. (1995). *The nature of statistical learning theory*. Springer, NY.