# OBJECT RECOGNITION AND FEATURE EXTRACTION FROM IMAGERY: THE FEATURE ANALYST® APPROACH

J. S. Blundell, D. W. Opitz

Visual Learning Systems, Inc., 1719 Dearborn, Missoula, Montana 59801 USA
sblundell@vls-inc.com, opitz@vls-inc.com

**KEY WORDS**: Feature Analyst, Automated Feature Extraction, Machine Learning, Geospatial Feature

## ABSTRACT:

Feature Analyst is a commercial feature extraction software system that leverages multiple object recognition attributes, using semi-automated to automated workflows, to accelerate the collection of features from imagery. In this paper we describe the key components of the Feature Analyst system including workflow, user interface, and modeling approach for delivering automated feature extraction (AFE) capability to users collecting geospatial intelligence from satellite and aerial imagery sources.

## INTRODUCTION

Until recently there were two types of approaches for identifying and extracting objects of interest in remotely sensed images: *manual* and *task-specific automated* approaches. The first approach involves the use of trained image analysts, who manually identify features of interest using various image-analysis and digitizing tools. Features are hand-digitized, attributed and validated during geospatial data production workflows. Although this is still the predominant approach to geospatial data production, it falls short of meeting government and commercial sector needs for three key reasons: (1) the lack of available trained analysts; (2) the laborious, time-consuming nature of manual feature identification; and (3) the high labor costs involved.

Because of these drawbacks, researchers since the 1970s have been attempting to *automate* the object recognition and feature extraction process from imagery. This has traditionally been done by writing a task-specific computer program (Jain 1984; McKeown 1993). However, these programs take an exceedingly long time to develop, requiring expert programmers to spend weeks or months *explaining*, in computer code, visual clues that are often trivially obvious to the human eye. In addition, the resulting handcrafted programs are typically large, slow, and complex. Most importantly, they are operational only for the specific task for which they were designed, typically failing when given a slightly different problem such as a change in spatial resolution, image type, surface material, geographic area, or season. Developing such programs is complicated by the fact that user interest varies significantly. While some task-specific automated approaches have been successful, it is virtually impossible to create fully automated programs that will address all user needs for every possible future situation.

The Feature Analyst approach to object-recognition and feature extraction overcomes these shortcomings by using inductive learning algorithms and techniques to model the feature-recognition process, rather than explicitly writing a software program (Maloof 1998; Burl 1998). The user gives the system (computer program) a sample of extracted features from the image. The system then automatically develops a model that correlates known data (such as spectral or spatial signatures) with targeted outputs (i.e., the features or objects of interest). The learned model then automatically classifies and extracts the remaining targets or objects. Feature models can be cached in a repository, known as the Feature Model Library, for later use and the accompanying workflow and metadata (information on spectral bandwidth, date and time stamp, etc.) can be used to quickly compose new models for changing target conditions such as geographic location or hour of day. This approach leverages the natural ability of humans to recognize objects in complex scenes.

### Learning Applied to Image Analysis

Nearly all modern vision systems rely on handcrafted determinations of which operators work best for an image and what parameter settings work best for those operators (Maloof 1998; McKeown 1996). Such operators not only vary across the desired object to be recognized, but also across resolutions of the same image. Learning in object recognition works by acquiring task-specific knowledge by

watching a user perform the tasks, then refining the existing knowledge based on the feedback provided by the user. Therefore, the parameters for these objects are tuned by the learning algorithm "on-the-fly" during the deployment of the algorithm. It is not surprising, therefore, that visual learning (Nayar and Poggio 1996; Ikeuchi and Veloso 1997) can greatly increase the accuracy of a visual system (Maloof 1998; Burl 1998).

In addition to increasing overall accuracy, visual learning can yield object-recognition systems that are much *easier* and *faster* to develop for a particular problem and resolution (Hepner 1990). A model can be trained for one task then used to "seed" the development of a similar problem. This allows the immediate deployment of a new problem with the ability to fine-tune and improve itself though experience. By having a learning system at the core of the object recognition task, one can easily transfer pertinent knowledge from one problem to another, even though that knowledge may be far from perfect. Prior research on visual learning has primarily consisted of hard-coded problem-specific learning models.

An *inductive learner* is a system that learns from a set of labeled examples. A teacher provides the output for each example, and the set of labeled examples given to a learner is called a *training* set. The task of inductive learning is to generate from the training set a concept description that correctly predicts the output of all future examples, not just those from the training set. Many inductive-learning algorithms have been previously studied (Quinlan 1993; Rumelhart 1986). These algorithms differ both in their concept-representation language, and in their method (or *bias*) of constructing a concept within this language. These differences are important because they determine the concepts that a classifier induces.

*Feature selection* in inductive learning is the important task of identifying the set of features (i.e., inputs) that are given to a learning algorithm. Automated feature selection is a central problem in inductive learning because there are numerous possible low-level image features (e.g., pixel values, edges, elevation, slope, etc.) that can be used by a learning algorithm. Rather than follow the traditional approach of finding one subset of features to give a learning algorithm VLS introduced the concept of *ensemble feature selection* to the problem of recognizing and extracting features from imagery (Opitz 1999). In this approach, the algorithm searches for a set of feature subsets that is not only germane to the learning task and learning algorithm, but that actually promotes *disagreement* among the ensemble's classifiers.

GEFS (for *Genetic Ensemble Feature Selection*) uses genetic algorithms (GAs) to generate a set of classifiers that are accurate and diverse in their predictions (Figure 1).

In Figure 1 neural networks are the classification method used (though conceptually any classification method can be substituted in place of the networks). Each network in the ensemble (networks 1 through *N*) is trained using the training instances for that network. Then, the predicted output of each of these networks is combined to produce the output of the ensemble (Ô in Figure 3). Both theoretical research (Opitz and Shavlik 1999; Shapire et al. 1997) and empirical work (Opitz and Maclin 1997; Opitz and Shavlik 1996a) have shown that a good ensemble is one in which (1) the individual networks are accurate, and (2) any errors that these individual networks make occur in different parts of the ensemble's input space. Much ensemble research has focused on how to generate an accurate, yet diverse, set of predictors. Creating such an ensemble is the focus of VLS's GEFS neural network ensemble algorithm (Opitz 1999). GEFS has proven to be more accurate on most domains than other current state-of-the-art learning algorithms (including Bagging and Boosting) and works particularly well on problems with numerous diverse inputs, such as high-resolution multispectral and hyperspectral images.

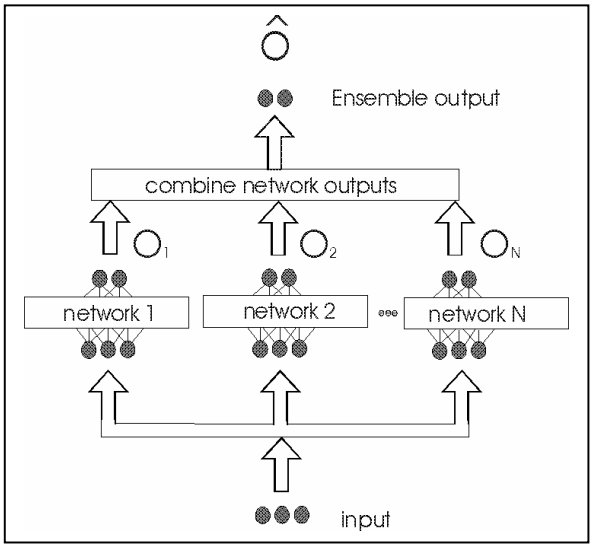


**Figure 1. Predictor Ensemble for Data Fusion.**

## FEATURE ANALYST

In 2001 Visual Learning Systems, Inc. (VLS) developed Feature Analyst as a commercial-off-the shelf (COTS) feature extraction extension for ArcGIS in response to a the geospatial market's need for automating the production of geospatial features from earth imagery. Feature Analyst uses an inductive learning based approach to object-recognition and feature extraction. As an extension to established systems Feature Analyst is designed, both from a workflow and user interface perspective, to integrate its

AFE approaches into a geospatial feature collection environment that is familiar to the customer. Benefits of this design include:

- **Significant time-savings** in the extraction of 2-D and 3-D geospatial features from imagery. NGA's InnoVision STAR Program studies indicate Feature Analyst is 5 to 10 times faster than manual extraction methods and more accurate than hand-digitizing on most features (O'Brien, 2003). (Figure 2)
- **Workflow extension** capabilities to established software ArcGIS, ERDAS IMAGINE, SOCET SET and soon Remote View. Analysts can leverage Feature Analyst within their preferred workflow on these existing systems which increases operator efficiency and output.
- **Simple *One-Button* approach** for extracting features using the Feature Model Library as well as advanced tools for creation of geospecific features from high resolution MSI, radar, LIDAR, and hyperspectral data.
- **Open and standards-based software architecture** allowing third-party developers to incorporate innovative feature extraction algorithms and tools directly into Feature Analyst.
- **Interoperability amongst users** on all three platforms. Expert analysts can create and store AFE models in the Feature Model Library, while other analyst can use these models for easy one-button extractions.

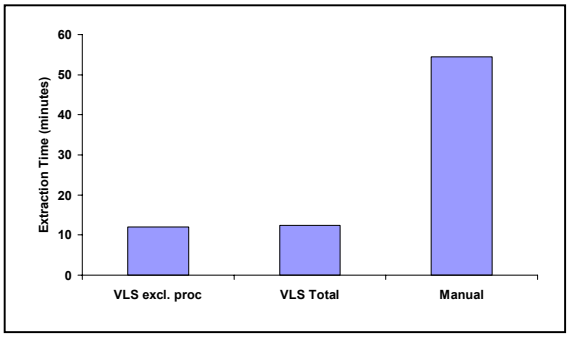


**Figure 2. NGA AFE Test and Evaluation Program timing comparisons (O'Brien, 2003).**

Key components of the Feature Analyst system for object recognition and feature extraction from imagery include the following:

- A simple User Interface (UI) which hides the complexity of the AFE approaches.
- State-of-the-art learning algorithms for object recognition and feature extraction.
- Post-processing tools for editing and generalizing features.
- AFE modeling tools for capturing workflows and automating feature collection tasks.

**Feature Analyst Workflow**

In the Feature Analyst system, the image analyst creates feature extraction models simply by classifying on the computer screen the objects of interest in a small subset of the image or images (Opitz and Blundell, 1999). This approach leverages the natural ability of humans to recognize complex objects in an image. Since the user does not require programming knowledge, users with little computational knowledge can effectively create visual models for the tasks under consideration. In addition, different users can focus on the different features of interest, with the system dynamically learning these features. This novel *assisted feature extraction* (AFE) approach leverages the natural ability of humans to recognize objects in imagery.

Figures 3-5 show the value of using spatial attributes, such as spatial association, in feature extraction. Figure 3 is the sample image. In this case, we want to extract white lines on airport runways. Using only spectral information, the best an analyst can do is shown in Figure 4; the results show all materials with similar "white" reflectance are extracted. Figure 5 shows the Feature Analyst results for extracting white lines using a spectral values and spatial parameters. In this case, one needs to know the neighboring pixels for the targeted white line are pavement or grass when extracting lines. This example illustrates the need to take into account spatial information when conducting true feature extraction.

**Figure 3. Sample of the original image – the objective is to extract only the thin white lines on the runway.**

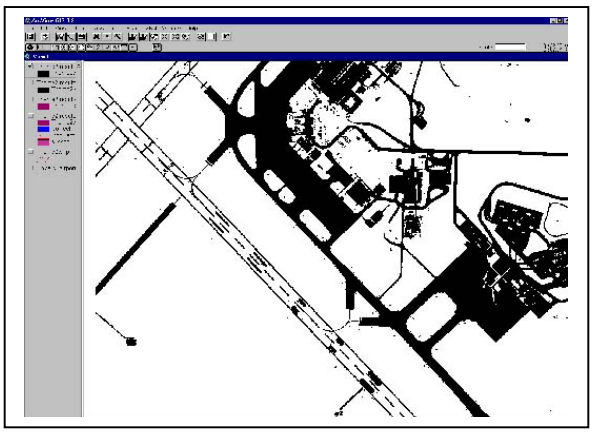

**Figure 4. Extraction results without the use of spatial attributes.**

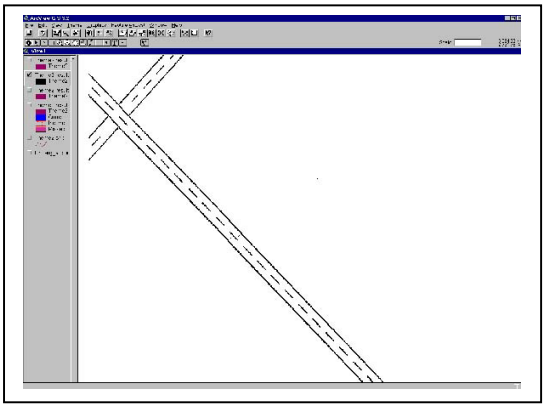

**Figure 5. Feature Analyst classification using spatial attributes to extract only the targeted white lines on the runway.**

The Feature Analyst workflow includes the following steps:

1. User digitizes several examples of the feature they are trying to collect. In the extraction example shown above in Figure 5, the user only had to digitize 3-4 labeled examples for the learning algorithm to extract all of the features correctly.
2. User selects the feature type from the UI which automatically sets all of the learning parameters behind the scenes.
3. User extracts features using the One-Button approach.
4. User examines results and, if required, provides positive and negative examples to remove clutter using Hierarchical Learning.

The goal of hierarchical feature extraction is to leverage a human's impressive vision ability to improve classification results by mitigating clutter (false positives), and retrieving missed objects. Clutter is the most common form of error in feature extraction. The objective of clutter mitigation is to remove false positives. Thus, the learning task is to distinguish between false positives and correctly identified positives. The user generates a training set by labeling the positive features from the previous classification as either positive or negative. The trained learner then classifies only the positive instances from the previous pass. The negative instances from the previous pass are considered correct in clutter mitigation and are thus masked out.

Hierarchical learning is necessary for learning complex targets. The overall process iteratively narrows the classification task into sub-problems that are more specific and well defined. The user begin the hierarchical process the same as any baseline inductive learning classification, i.e., select labeled examples for the feature being extracted, train the learner, and then classify every pixel in the image based on the learner's prediction. At this point if the user is not satisfied with the results, they can apply a hierarchy of learners to improve the classification. The classification is improved in passes where each new pass is designed to remove one form of error from the results of the previous pass.

**Automation of Feature Extraction Tasks**

Automated feature extraction has been the long-term goal of geospatial data production workflows for the past 30 years. The challenge is developing a flexible approach for transferring domain knowledge of a feature extraction model from image to image that is capable of adapting to changing conditions (image resolution, pixel radiometric values, landscape seasonal changes, and the complexity of feature representation). In 2006 VLS introduced the concept of Feature Modeler and the Feature Model Library (FML) as tools within Feature Analyst to automate the feature extraction workflow process. Feature Modeler provides users with a comprehensive set of tools for examining and refining feature models created with Feature Analyst (Figure 6). Feature models, designated as AFE models, include the parameter settings for a classifier to extract a particular feature including spatial context, and hierarchical learning passes. Benefits of this approach include the following:

- Analyst can create, edit and refine the inner workings of an AFE model including the pixels used for classification of a feature with spatial processing, priority of input bands, rule extraction from a complex learned model and the parameter settings for a learning algorithm.
- Technicians can access AFE models and run them in an interactive mode or run the model in a silent batch

mode. In interactive mode the technician doesn't need to worry about creating the proper workflow or setting parameters for the learning algorithm; rather they only have to provide a labeled set of examples. In batch mode the process is completely automated where a single AFE model, or multiple AFE models, can be run against a single image or a directory of images,

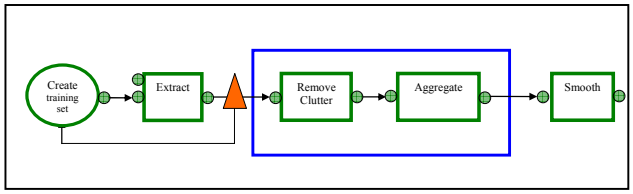


**Figure 6. A simple AFE model representation showing the five steps used in processing. Each of the processing steps can be adjusted by a user and the model can be re-used for a different image.**

The Feature Model Library (FML) resides within a relational database and is used for storing AFE models to support enterprise-wide geospatial processing. Analysts can search and retrieve AFE models for subsequent processing in a batch mode to extract features from imagery without any training sets. The Feature Modeler software allows a user to import an AFE model to examine and adjust parameter settings or deploy a learning model in a batch-processing mode.

## CONCLUSIONS

1. Feature Analyst provides a comprehensive system for assisted and automated feature extraction using earth imagery in commercial GIS, image processing and photogrammetry software. The AFE workflow is integrated with the supporting application tools and capabilities which provides a more holistic solution for geospatial data production tasks.
2. The user interface supports a simple feature extraction workflow whereby the user provides the system with a set of labeled examples (training set) and then corrects the predicted features of the learning algorithm during the clutter removal process (hierarchical learning).
3. The Feature Analyst approach uses inductive learning and the incorporation of object-recognition attributes to extract features using a feature model.
4. Feature models are adaptive and can be modified to extract features from imagery collected at different geographic locations, resolutions, etc. The Feature Modeler provides tools for examining and refining feature models which can then be shared with other users. This approach captures a user's workflow and allows models to be re-used and shared within an enterprise.

## REFERENCES


Burl et al. 1998. "Learning to Recognize Volcanoes." *Machine Learning Journal* 30 (2/3), 165-194.

Hepner et al. 1990. "Artificial Neural Network Classification Using a Minimal Training Set: Comparison to Conventional Supervised Classification." *Photogrammetric Eng. and Remote Sensing* 56: 469-473.

Ikeuchi, K., Veloso, M. 1997. *Symbolic Visual Learning*. New York, NY: Oxford University Press.

Jain, R. et al., 1995. *Machine Vision.* McGraw-Hill, New York, NY.

Maloof et al. 1998. "Learning to Detect Rooftops in Aerial Images." *Image Understanding Workshop* 835-845. Monterrey, CA.

McKeown et al., 1993. "Research in Automated Analysis of Remotely Sensed Imagery" *Proceedings of the DARPA Image Understanding Workshop*, Washington DC, April 18-21, 1993.

McKeown, D. 1996. "Top Ten Lessons Learned in Automated Cartography," *Technical Report CMU-CS-96-110*, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

Nayar, S., Poggio, T, eds, 1996. *Early Visual Learning*. New York, NY: Oxford University Press.

Opitz, D., Maclin, R. 1997. "An Empirical Evaluation of Bagging and Boosting." Proceedings of the *14th National Conference of Artificial Intelligence*, Providence (AAAI), 546-551.

Opitz, D., Maclin, R. 1999. "Popular Ensemble Methods: An Empirical Study." *Journal of Artificial Intelligence Research*. 11 (1): 169-198.

Opitz, D., Shavlik, J. 1996a. "Generating Accurate and Diverse Members of a Neural-Network Ensemble." *Advances in Neural Information Processing Systems* 8: 535-541.

Opitz, D. 1999. "Feature Selection for Ensembles." *Proceedings of the 16th National Conference on Artificial Intelligence*. 379-384.

Opitz, D., Bain, W. 1999. "Experiments on Learning to Extract Features from Digital Images" *IASTED Signal and Image Processing*. In press.

Opitz, D., Blundell, S. 1999. "An Intelligent User Interface for Feature Extraction From Remotely Sensed Images." *Proc. of the American Society for Photogrammetry and Remote Sensing*, 171-177.

Opitz, D., Shavlik, J. 1999. "Actively Searching for an Effective Neural-Network Ensemble." *Springer-Verlag Series on Perspective in Neural Computing*. 79-97..

Quinlan, J. 1993. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Rumelhart et al. 1986. "Learning Internal Representations by Error Propagation." In: *Distributed Processing: Explorations in the Microstructure of Cognition*. Volume 1: 318-363.

Shapire, R., Freund, Y. 1997. *Proc. of the 14th International Conference on Machine Learning* 322-330.