Online Feature Selection for Pixel Classification

Karen Glocer KAG@CS.UCSC.EDU
Department of Computer Science, University of California Santa Cruz

Damian Eads EADS@LANL.GOV
James Theiler JT@LANL.GOV
Los Alamos National Laboratory, Los Alamos, NM 87544 USA

Abstract

Online feature selection (OFS) provides an efficient way to sort through a large space of features, particularly in a scenario where the feature space is large and features take a significant amount of memory to store. Image processing operators, and especially combinations of image processing operators, provide a rich space of potential features for use in machine learning for image processing tasks but they are expensive to generate and store. In this paper we apply OFS to the problem of edge detection in grayscale imagery. We use a standard data set and compare our results to those obtained with traditional edge detectors, as well as with results obtained more recently using "statistical edge detection." We compare several different OFS approaches, including hill climbing, best first search, and grafting.

1. Introduction

Traditional learning systems assume that all features are readily available from the beginning, but there are scenarios where not all features are present initially and must be integrated as they become available. The online feature selection (OFS) problem is formulated with a training set consisting of n pairs (x, y) of instances and labels. Typically, $x \in R^d$ and, for binary classification, $y \in \{-1, 1\}$. No new instances are added so n remains constant, but d is not. Given a way to integrate the new features as they arrive, the computation can begin with a small set of features and more features can be added as they become available.

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

The primary contribution of this paper is the application of OFS algorithms to a real-world problem in a domain where features truly are generated online. In this domain, where the potential feature space is enormous but for which only a few features can be held in memory at any given time, OFS is a necessity rather than a luxury. While the feature selection literature is ample, the OFS literature is rather sparse and mostly involves example problems where all of the features are actually available beforehand (Perkins & Theiler, 2003). In this paper we demonstrate the power of OFS in the image processing domain by applying it to the problem of edge detection. We think this is a natural fit for OFS and show that it provides an approach that can outperform existing algorithms for edge detection.

An edge may be described informally as the boundary between adjacent parts of an image, but a formal definition is elusive. Different applications have different requirements, and these requirements are often informal as well. For example, both image segmentation and vectorization use edge detection as an intermediate step, but the edges that are optimized for one purpose may not be as useful for other purposes. This is a compelling argument for using learning techniques to design edge detectors. Machine learning algorithms do not require a formal definition of edges; instead they take examples as input – marked up images that identify where the edges are – and from those, produce an algorithm that finds edges in new images.

In the image processing domain, each pixel in an image is an instance and for the edge detection problem, each label is either on-edge or off-edge. We remark that pixels are spatially related to their neighbors and therefore are not iid. This not necessarily a drawback because it is this spatial relationship between pixels that we exploit to identify edges.

In this work we deal with two component problems: feature extraction and feature selection. We begin with training images and a corresponding ground truth that marks each pixel in the training image as either on-edge or off-edge. Our feature extractor, described in Section 2, generates combinations of image processing operators and applies them to the training images to produce features. A classifier is then produced from a linear combination of these features. The online feature selection algorithm, discussed in Section 3, selects a subset of those features and integrates them into the model as they arrive. The subset of features that was selected is stored and d more features are generated and added to the set. Then the online feature selection algorithm selects another subset. The last two steps are repeated until a stopping condition is met.

2. Feature Extraction

Feature extraction is the problem of finding alternate representations of the underlying data. In image processing, a feature can be generated as the output of one of a series of operators applied to an image. The scalar value of a pixel in the output image is the value of the feature for that instance. Feature extraction encompasses not only the question of which operators are used but also the complexity with which they are combined. We study the effects of feature complexity by comparing simple features that consist of a single operator to features whose structure is a tree of operators.

2.1. Simple Features

A "simple" feature is generated by running a single image processing operator on a raw input image. The operators used in this work are listed in Table 1. Most of these operators are neighborhood operators, where each output pixel is computed from a set of neighboring input pixels.

For example, each pixel in the output of the Gaussian smoothing operator is a weighted average of the pixels in its neighborhood in the input image. The weights are given by a Gaussian centered at that pixel. The gradient operators produce output that is based on the differences between pixels and its neighbors. The statistical operators perform various computations (*e.g.,* minimum, maximum, or standard deviation) on the pixel values in a neighborhood of a given pixel and the result of those computations provide the pixel values for the ouput image.

Non-maximal suppression and hysteresis thresholding are specialized operators that were added because they are particularly valueable in edge detection.

Gabor filters are commonly used for texture recognition and other low level image processing tasks. They possess good localization properties in both the spatial and frequency domains. Gabor functions are complex exponentials modulated by a Gaussian envelope. The implementation we used was a multi-resolution pyramidal filter bank for MATLAB designed by Nestares et al. (1998). To achieve rotational invariance, we interpolate the oriented components for each scale of the pyramid of filters using the Discrete Fourier Transform (Greenspan et al., 1994).

Table 1. These are the operators used to produce simple features. In addition, they are the building blocks for all tree-structured features. All of these features are grouped into categories so that when we discuss more complex features, the representation can be more compact.

Feature	Category
Gaussian Smoothing	Smooth
2-D Gradient	Grad
Sobel Gradient	Grad
Min, Max, Peak	Stat
Standard deviation	Stat
Non-maximal suppression	Stat
Hysteresis thresholding	Stat
Morphological open, close	Morph
TopHat	Morph
Gabor filter	Gabor
Rotation Invariant Gabor	Gabor

2.2. Tree-Structured Features

Simple features used with a linear classifier restrict the hypothesis class to linear combinations of single operators. Unfortunately, this class is not rich enough to learn hypotheses that require combinations of operators. For instance, the well-known Canny (1986) edge detector runs the following operations sequentially: Gaussian smoothing, 2-D gradient, non-maximal suppression, and hysteresis thresholding. If we restrict ourselves to simple features, we would be trying to learn Canny by taking the linear combination of a smoothed image, a gradient image, a non-maximally suppressed image, and a thresholded image. To make matters worse, those last two operators make very little sense unless they are run in concert with other operators.

Allowing features to have a more complex structure produces a richer hypothesis class. We used tree structures because they have been demonstrated to work well in other situations (Koza, 1992). Given a rich set of operators, these trees can represent a large variety of functions of the original input data. The operators

Table 2. This is the grammar used to generate features. In addition to this root production, a feature can also be generated by adding, subtracting, multiplying, or dividing any two features generated from these rules.

RootProduction ::=
Morph (Stat (Grad (Gabor (Smooth (x))))) |
Morph (Stat (Grad (Smooth (x)))) |
Morph (Stat (Gabor (Smooth (x)))) |
Morph (Stat (Smooth (x))) |
Morph (Grad (Gabor (Smooth (x)))) |
Morph (Grad (Smooth (x))) |
Morph (Gabor (Smooth (x))) |
Morph (Smooth (x)) |
Morph (Gabor (x)) |
Morph (Morph (x)) |
Morph (Stat (x)) |
Morph (Grad (x)) |
Morph (x) |
Stat (Grad (Gabor (Smooth (x)))) |
Stat (Gabor (Smooth (x))) |
Stat (Grad (Smooth (x))) |
Stat (Smooth (x)) |
Stat (Gabor (x)) |
Stat (Grad (x)) |
Stat (Stat (x)) |
Stat (x) |
Grad (Gabor (Smooth (x))) |
Grad (Gabor (x)) |
Grad (Smooth (x)) |
Grad (x)

for these trees are the same as those used to generate simple features, but are combined only in ways that make sense given the nature of the operators. This constraint is provided by a context-sensitive grammar. The advantages of grammars are threefold. First, only sensible features are generated. Second, by restricting the way in which features can be combined, grammars greatly reduce the size of the search space. Third, they are a provide an effective way to incorporate domain knowledge. The root production of the grammar is described in Table 2.

3. Online Feature Selection

The online feature selection problem assumes that features arrive in stages but that no new instances are added to the problem. At stage t, a new set f_t of features arrives. The set of all features at stage t is denoted by F_t. Thus $F_t = \{f_t \cup F_{s,t-1}\}$, the union of features that have just arrived with the set of features that was selected at time $t-1$. At time t , after the arrival of f_t, a feature selection algorithm selects the subset $F_{s,t} \subseteq F_t$ based on some as yet unspecified criterion. In this way online feature selection can be viewed as adding a wrapper around a feature selection algorithm that is parameterized by the number of features added per stage, $d_t = |f_t|$, and by the feature selection algorithm it uses. Feature selection algorithms generally fall into three main categories: filters, wrappers and embedded methods (Guyon & Elisseeff, 2003). In this paper we use two wrappers and one embedded method: hill climbing, best first search, and grafting.

3.1. Hill Climbing

The hill climbing algorithm initializes a cache with d features. The initial fitness of the algorithm is the fitness of the initial cache. The cache is then mutated in one of three ways, each of which are equally probable: a randomly selected feature is removed from the cache, a randomly generated feature is added to the cache, or a randomly selected feature in the cache is replaced with a randomly generated feature. If this mutation improves the fitness, the mutation is kept. Forrest and Mitchell (1993) refer to this as random mutation hill climbing, or RMHC. The criterion we used to evaluate the fitness of the cache is the "empirical Bayes risk," R_{Bayes}, which can be computed from the ROC curve by finding the point on the curve where the slope is 45 degrees (Green & Swets, 1966).

3.2. Best First Search

Best first search, according to Kohavi and John (1997), is more robust than hill climbing. It is, in any case, a more systematic approach to feature subset selection. The two most common variants of best first search are sequential forward selection and sequential backward elimination. Although it can be slower, we use backward elimination because it is not sensitive to the feature that is chosen first. The number of features per stage of the online feature selection algorithm is small enough that the computation time for either algorithm is much less than the time it takes to extract new features.

Sequential backward elimination is initialized with a full set of features F_0. The algorithm first removes each feature $f_i \in F_0$ from F_0 and trains an induction algorithm with the feature set $F_0 - \{\mathbf{f}_i\}$. The feature $\mathbf{f}_{max}$ whose removal minimizes the error in the resulting classifier is removed, and $F_1 = F_0 - \{\mathbf{f}_{max}\}$. In the next round, for each of the remaining features $\mathbf{f}_i \in F_1$, the algorithm tests the feature subset $F_1 - \{\mathbf{f}_i\}$ and removes the feature $\mathbf{f}_{max}$ that minimizes the error of the resulting classifier to produce $F_2 = F_1 - \{\mathbf{f}_{max}\}$. This process repeats until a local minimum of classifier error has been reached or some other stopping condition is met.

In practice, removing a feature may have only a very small negative impact on performance, so backward elimination becomes a trade-off between marginally lower performance and a smaller set of features. This implies a need for some sort of regularization. Following Kohavi and John (1997), we added a penalty $c = 0.001$ per feature – or equivalently, the zero-norm of the weight vector (Weston et al., 2003) – to force the algorithm to favor smaller subsets. Thus our best first search algorithm minimizes the loss function

$$L = R_{\text{Bayes}} + c|w|_0.$$

where R_{Bayes} is the empirical Bayes risk, described earlier.

3.3. Grafting

Grafting (Perkins et al., 2003) recasts feature subset selection as the minimization of a regularized risk criterion of the form:

$$C = L(f(x)) + \lambda \sum_{j=1}^{d} |w_j|.$$

The second term, the regularizer, is the ℓ_1 norm of the weight vector. The free parameter, λ, characterizes the trade-off between accuracy and complexity. The first term of the criterion function is the loss function, which in our implementation is the binomial negative log likelihood (BNLL) loss described in Hastie et al. (2001):

$$L(f(x)) = \frac{1}{n} \sum_{i=1}^{n} \ln(1 + e^{-y_i f(x_i)}).$$

The grafting algorithm is based on the observation that the addition of feature i incurs a penalty of $\lambda|w_i|$. Thus adding the feature is only worthwhile if the reduction of the mean loss is greater than the increase in the penalty, and that this will only happen if $|\partial L/\partial w_j| > \lambda$. This gradient test is performed for each feature as it arrives, and it is faster than re-optimizing the classifier with respect to the new feature. If no weights pass the test, the feature is discarded. If at least one weight passes, then the weight that maximizes the magnitude of the gradient is added and the model is optimized with respect to all of its parameters. Grafting differs from the other algorithms because it considers not only whether to add a new feature but also whether to drop currently selected features and even whether to add discarded features.

4. Experimental Methodology

4.1. Data and Markup

We use the data set originally developed by Bowyer et al. (2001) in their extensive evaluation of various edge detectors and also used by Konishi et al. (2003) in their statistical edge detection work. The data set consists of 50 images and 50 hand-marked ground truths. To investigate the flexibility of our algorithm, we generated a second ground truth by running the Canny edge detector on the original data. The parameters of the Canny ground truth were the MATLAB defaults. Canny provides a consistent, deterministic ground truth, in contrast to the South Florida ground truth which is marked up by hand. All experiments were performed with both the original South Florida and the Canny ground truth.

4.2. Methodology

Every experiment used a training set of four images selected at random and a validation set consisting of the remaining 46 images in the South Florida data set. There are a great deal more off-edge pixels than on-edge pixels, so to balance the training set we subsampled the four training images. The subsampled training set consists of all edge pixels and an equal number of randomly selected off-edge pixels. The training set contained approximately 200,000 pixels and the validation set contained approximately 14,000,000 pixels. To ensure that no feature dominates because of its initial scale, each feature was normalized to have a mean of zero and a standard deviation of one.

Simple features consist of a single operator applied to the input data. There are 14 operators, shown in Table 1. Even with a range of different parameter values, the total number of simple features was only 80. With such a small number of features, random mutation hill climbing was not used.

The space of tree-structured features is much larger than the space of simple features; not every possible feature will be searched. A fair evaluation of these algorithms requires that each algorithm be given the same resources. Because generating the features is the time-limiting step, we allowed each algorithm to explore a total of 2000 features.

The hill climber was initialized to one feature. At each iteration, a feature was either added to the cache, removed from the cache, or replaced with another feature. The mutation was kept if it improved fitness. Although the mutation of the feature cache was selected randomly, we kept track of how many times a feature was added or replaced to ensure that the hill climber was able to explore 2000 features.

With best first search, we initialized the feature cache to ten features. We then ran backward elimination with a complexity penalty of $c = 0.001$ per feature. The remaining subset of features was evaluated with Fisher's linear discriminant. In the next stage, ten new features would be added to the previous set and backward elimination would again be applied. In this way 2000 features are explored in 100 iterations.

Grafting differs from the other algorithms in that whenever it adds a new feature, it can check discarded features and potentially reincorporate them into the model. The best result would probably come from keeping all 2000 features in memory, but this is not feasible. Instead, we do 40 iterations that explore 50 features at a time.

4.3. Performance Metrics

For edge detection, most evaluation is based on agreement with manually determined ground truth, and this agreement is usually measured in terms of ROC curves (Zhu, 1996; Shin et al., 1999; Forbes & Draper, 2000; Konishi et al., 2003). When the output of an edge detector is compared to a ground truth value, a count of edge detections and false alarms can be obtained. Comparing two edge detectors knowing only these quantities can be ambiguous. It is hard to say whether it is better have a high detection rate or a low false alarm rate. By adaptively exploring the parameter space of an algorithm and plotting the resulting detection and false alarm rates against each other, a ROC curve can be used to describe the performance of an edge detector more completely. For the ROC curves reported here, we sweep over the value of the threshold and compute a detection rate and false alarm rate for each value.

4.3.1. Pixel-to-pixel metric

For each image in our data set, there is a corresponding ground truth image with each pixel is divided into two classes: on-edge and off-edge. Konishi et al. (2003) compare every pixel from the output of their edge detector to the corresponding pixel in the ground truth. An error occurs when the output pixel does not match the ground truth pixel. The detection rate (DR) and false alarm rate (FA) are then given by

$$\begin{aligned} DR &= \frac{N_{(\text{on-edge} \mid \text{on-edge})}}{N_{\text{on-edge}}} \\ FA &= \frac{N_{(\text{on-edge} \mid \text{off-edge})}}{N_{\text{off-edge}}}. \end{aligned}$$

4.3.2. Bowyer metric

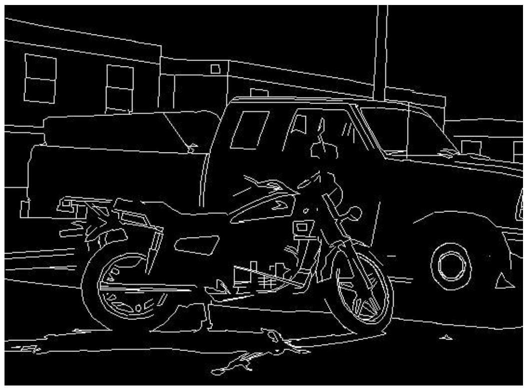
We also used a second metric so we could compare our results to those of Bowyer et al. (2001). The original ground truth for the South Florida data set was actually three-valued. Pixels are either on-edge, off-edge, or "neutral". The neutral pixels, which are usually near the true on-edge pixels, are those for which misclassification is not penalized. Since this is an area where mistakes are easy for a classifier to make, deweighting these points generally leads to higher performance scores regardless of whether the performance itself is actually improved. The neutral category reduces the false alarm rate and leaves the detection rate unchanged, shifting the ROC curve to the left. But this is also the region where mistakes are less problematic, so this metric might more accurately reflect what a practical edge detector is trying to do.

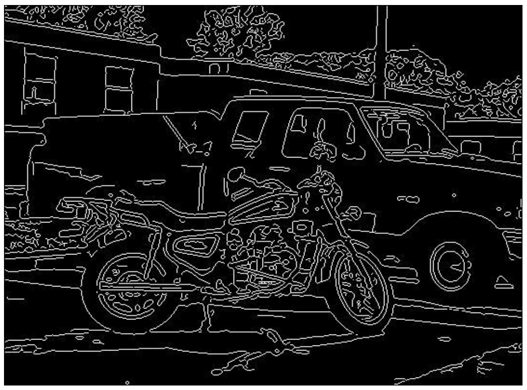
The Bowyer metric works like this: if a pixel is classified as an edge pixel but it falls in a background region, it is counted as a false positive; if it falls on an edge pixel or within a radius T_{match} of an edge, it is counted as a true positive. To ensure that no ground truth pixel is counted twice, if a pixel falls within T_{match} of the ground truth pixel, the ground truth pixel is marked so that it cannot be counted again. T_{match} is an adjustable tolerance parameter. Note that this metric is applied to the ground truth data with neutral pixels, and that these neutral pixels provide a buffer of size at least T_{match} between the on-edge and off-edge pixels in the ground truth images. Bowyer uses a value of $T_{\text{match}} = 3$ and for consistency, we use the same value.

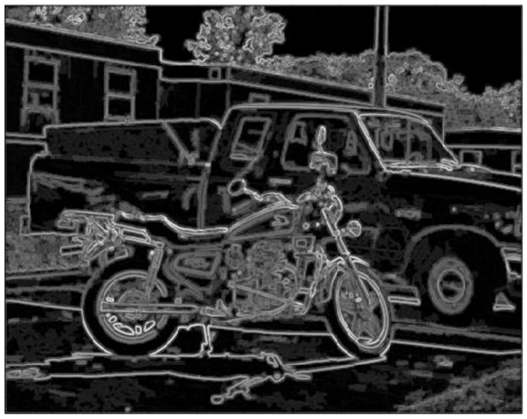
5. Results

In this section, we analyze the performance of different feature selection algorithms and compare them to the previous work of Bowyer et al. (2001), who did an extensive comparison of classical edge detectors, and to Konishi et al. (2003), who used an adaptive grid to model the statistical distribution of on-edge versus off-edge pixels and applied maximum likelihood to this estimate of the distribution.

To illustrate the nature of these experiments, Figure 1 shows the South Florida ground truth and the Canny ground truth for an image in the South Florida data set. The South Florida ground truth is actually three-valued but by considering the neutral regions to be off-edge, we have condensed it down two classes: the white pixels are on-edge and the black pixels are off-edge. Below each ground truth is the grayscale result of our algorithm trained on that ground truth using tree-structured features and grafting. The grayscale


(a) South Florida Ground Truth


(b) Canny Ground Truth


(c) Result for South Florida Ground Truth

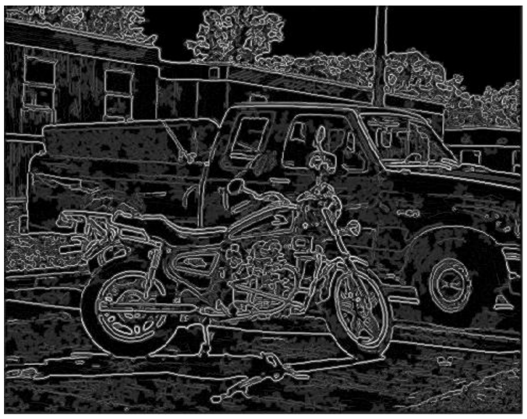

(d) Result for Canny Ground Truth

Figure 1. South Florida and Canny ground truths and their respective results.

value of a pixel is its decision value before thresholding. The results are visibly different from each other but similar in appearance to their respective ground truths.

Qualitative visual comparisons are no substitute for quantitative comparisons, so we scored each result image in Figure 1 against both ground truths. To get a single numerical value for a score, we constructed ROC curves for the comparison between each result and ground truth, found empirical Bayes risk, and turned it into score $S = \frac{1}{2}(DR + (1 - FA))$. When evaluated on this South Florida ground truth image, the algorithm trained on the South Florida ground truth (which not include this image) scored a 0.8 while the algorithm trained on the Canny ground truth scored a 0.78. Similarly, when evaluated on the Canny ground truth image in Figure 1(b), the algorithm that was trained on the Canny ground truth scored a 0.88 while the algorithm that was trained on the South Florida ground truth only scored a 0.83. This demonstrates the flexibility of the machine learning approach.

There is a definite advantage to increasing the complexity of the feature structure. When tree-structured features were compared to simple features, the tree-structured features invariably outperformed the simple features for the same algorithm. Figure 2 shows results for simple features for grafting and best first search. For comparison, results for tree-structured grafting are included as well. The difference is dramatic. Also included are results for statistical edge detection. Note that even simple features produce results that are comparable to statistical edge detection.

A more quantitative evaluation of tree-structured features on the pixel-to-pixel metric is shown in Figure 3.

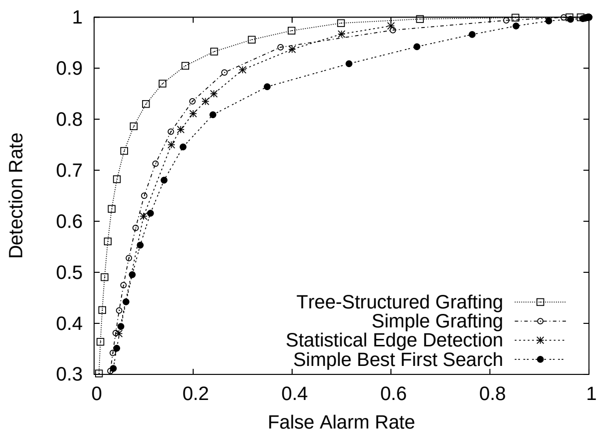


Figure 2. Simple features for grafting and best first search scored with the pixel-to-pixel metric and compared with statistical edge detection.

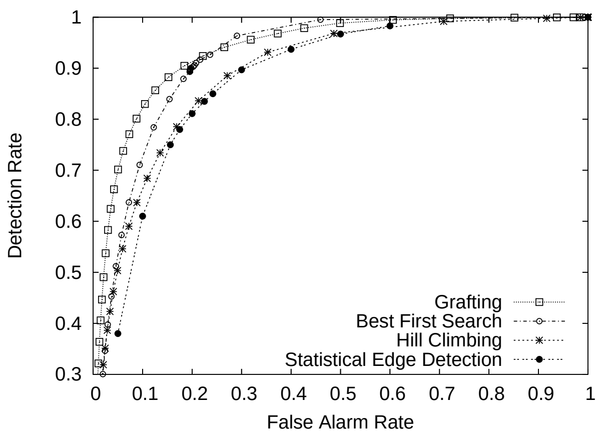


Figure 3. The performance of tree-structured features evaluated using South Florida ground truth scored with the pixel-to-pixel metric.

It shows ROC curves for grafting, best first search, and hill climbing. It also includes results for statistical edge detection (Konishi et al., 2003) because it was evaluated on the same ground truth and the same metric. Grafting outperforms best first search, which outperforms hill climbing. Hill climbing, in turn, was slightly better than statistical edge detection.

In principle, our approach should be able to find a feature that is a serial application of four operators (Gaussian smoothing, 2-D gradient, non-maximal suppression, and hysteresis thresholding) to give an exact match to the Canny ground truth, but the probability of generating the exact feature is minuscule. Instead, the algorithm found a set of 17 features whose linear combination approximates Canny. This result is significant because it demonstrates that even if the ideal feature is never found (supposing it exists in the first place), similar but imperfect features can be combined to yield a good classification. Figure 4 shows ROC curves for grafting, best first search, and the hill climber all trained on the Canny ground truth. Grafting is marginally better than best first search and significantly better than hill climbing. There is no direct comparison to statistical edge detection because of the difference in ground truth.

Finally, scoring our results with the Bowyer metric allows us to compare these results directly with those of Bowyer. Figure 5 compares grafting results with the Canny edge detector in Bowyer et al. (2001). The change in metric causes the score of the results to improve tremendously even if the classifier itself changes very little when trained on this metric.

6. Conclusion and future work

We have demonstrated the utility of online feature selection for one of the most useful and ubiquitous tasks in image processing: edge detection. Image processing is a domain where the computational bottleneck is the feature generator, not the classifier. Adding complexity to the structure of the features makes the feature space too large to search exhaustively but it increases the flexibility of the hypothesis class and can improve the performance of the resulting classifier. Online feature selection provides a systematic way to search this larger feature space, and we show that tree-structured features and online feature selection techniques outperform previous work by a significant margin. Grafting in particular is found to be a fast, efficient way to incorporate features in an online fashion. Furthermore, the online feature selection approach is flexible enough to perform well on different markups and across performance metrics.

In the future we would like to expand the online feature selection approach to image segmentation, of which edge detection is a component problem. Martin et al. (2004) argue convincingly that a good approach to boundary detection is to combine both edge and texture cues and they present an extensive benchmark. In contrast to Martin's approach, which focuses on the design of specific features, we consider the machine learning aspect of the problem and we argue that a broad, systematic search of less carefully tuned features has much to offer.

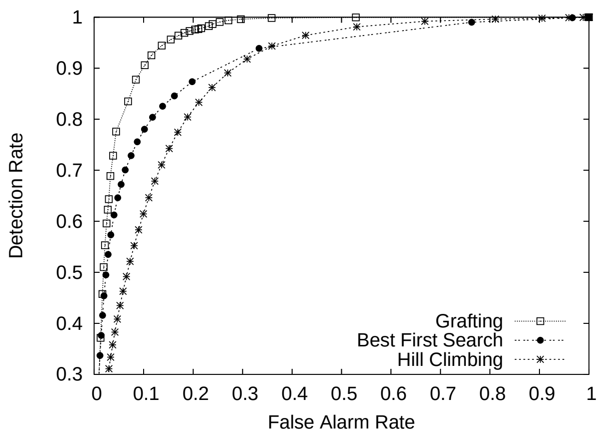


Figure 4. The performance of tree-structured features using Canny ground truth scored with the pixel-to-pixel metric.

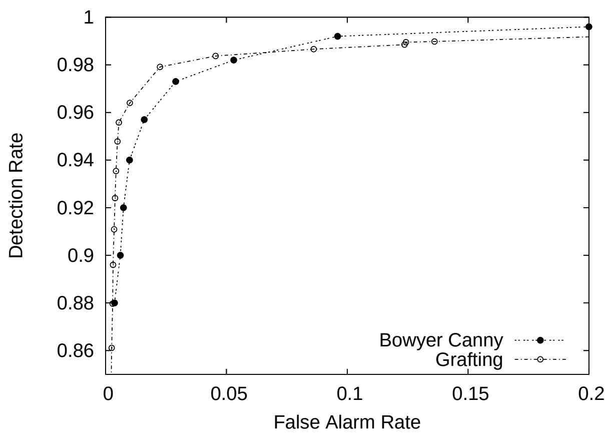


Figure 5. Grafting and Bowyer's implementation of Canny with tree-structured features on the South Florida ground truth scored with the Bowyer metric. For comparison, we also show the results of Canny from Bowyer et al. (2001).

Acknowledgments

We are grateful to Simon Perkins for many useful discussions, and for introducing us to online feature selection and to grafting.

References

Bowyer, K., Kranenburg, C., & Dougherty, S. (2001). Edge detector evaluation using empirical roc curves. *Computer Vision and Image Understanding*, *84*, 77–103.

Forbes, L. A., & Draper, B. A. (2000). Inconsistencies in edge detector evaluation. *Computer Vision and Pattern Recognition* (pp. 398–404). Hilton Head, SC.

Forrest, S., & Mitchell, M. (1993). Relative building-block fitness and the building-block hypothesis. In L. D. Whitley (Ed.), *Foundations of Genetic Algorithms 2*, 109–126. San Mateo, CA: Morgan Kaufmann.

Green, D., & Swets, J. (1966). *Signal Detection Theory and Psychophysics*. New York: Wiley.

Greenspan, H., Belongie, S., Perona, P., Goodman, R., Rakshit, S., & Anderson, C. (1994). Overcomplete steerable pyramid filters and rotation invariance. *CVPR94* (pp. 222–228).

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning*. New York: Springer.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*, 273–324.

Konishi, S., Yuille, A., & Coughlan, J. (2003). Statistical edge detection: Learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*.

Koza, J. (1992). *On the Programming of Computers by Means of Natural Selection*. Cambridge: MIT Press.

Martin, D. R., Fowlkes, C. C., & Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*, 530–549.

Nestares, O., Navarro, R., Portilla, J., & Tabernero, A. (1998). Efficient spatial-domain implementation of a multiscale image representation based on Gabor functions. *Journal of Electronic Imaging*, *7*, 166–173.

Perkins, S., Lacker, K., & Theiler, J. (2003). Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, *3*, 1333–1356.

Perkins, S., & Theiler, J. (2003). Online feature selection using grafting. *ICML* (pp. 592–599).

Shin, M. C., Goldgof, D. B., & Bowyer, K. W. (1999). Comparison of edge detectors using an object recognition task. *Computer Vision and Pattern Recognition* (pp. 360–365).

Weston, J., Elisseeff, A., Schlkopf, B., & Tipping, M. (2003). Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, *3*, 1439–1461.

Zhu (1996). Efficient evaluations of edge connectivity and width uniformity. *Image and Vision Computing*, *14*, 21–34.