Pattern Recognition 45 (2012) 241–251

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Self-tuned Evolution-COnstructed features for general object recognition

Kirt Lillywhite, Beau Tippetts, Dah-Jye Lee *

Department of Electrical and Computer Engineering, Brigham Young University, 459 Clyde Building, Provo, UT 84602, USA

A R T I C L E I N F O

Article history:
Received 23 May 2010
Received in revised form
28 April 2011
Accepted 24 May 2011
Available online 2 June 2011

Keywords:
Object detection
AdaBoost
Genetic algorithm
Feature construction
Self-tuned

A B S T R A C T

Object recognition is a well studied but extremely challenging field. We present a novel approach to feature construction for object detection called Evolution-COnstructed Features (ECO features). Most current approaches rely on human experts to construct features for object recognition. ECO features are automatically constructed by uniquely employing a standard genetic algorithm to discover multiple series of transforms that are highly discriminative. Using ECO features provides several advantages over other object detection algorithms including: no need for a human expert to build feature sets or tune their parameters, ability to generate specialized feature sets for different objects, no limitations to certain types of image sources, and ability to find both global and local feature types. We show in our experiments that the ECO features compete well against state-of-the-art object recognition algorithms.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Object recognition is a very challenging task that many researchers are currently pursuing [1]. The difficulty of detecting and labeling objects in images is due in part to issues such as lighting conditions, object pose, distortions, as well as naturally varying parameters of the objects. Overcoming these obstacles in order to achieve robust object recognition would be beneficial for many scientific fields and applications.

One effective, and thus popular, method for object recognition is to use machine learning to classify between object and non-object classes. In general, machine learning algorithms take in symbols, find patterns in the symbols, and use mathematical methods to separate the examples into classes. On the other hand, one of the main goals of computer vision is to take raw sensor data, the input signal, and create a set of symbols that represents the data. While research in the machine learning community continues to make significant progress, much of the problem with object recognition lies in the difficult mapping of signal to symbol. This is where we focus our efforts.

In object recognition these symbols are referred to as features. Machine learning techniques are successful when the set of features is able to uniquely describe the object of interest. Presently, human experts construct a feature set. Generally, this feature set comes with a host of parameters that can significantly alter the effectiveness of the feature set and also have to be tuned by the human expert. Parameter tuning can be a time consuming and very difficult task. One of the main challenges with classification algorithms is setting the parameter values to ensure good generalization performance [2]. Even with classification algorithms that have only a couple parameters, such as support vector machines, finding the optimal value of the kernel-parameter is expensive in terms of computing time [3]. The human expert is limited by time and can only try a small number of hand crafted solutions. Trial and error becomes his mantra with success coming here and there.

One successful object recognition algorithm, the histograms of oriented gradients (HOG) method, was introduced by Dalal and Triggs [4] for human detection. Many have based their feature set for object recognition on this method with various modifications or additions [5–8]. The HOG method captures local shape information by creating histograms of gradient orientations over a small region.

The work of Dalal and Triggs has been a very helpful and meaningful contribution to the scientific community but their work highlights the efforts that have to be made by a human expert in order to create a feature set effective for object recognition. A good portion of their paper outlines the parameters of the HOG algorithm and the systematic method they followed for finding the parameters that maximized their accuracy. They explore color normalization but abandon it when it provides no improvement. Smoothing methods are explored and then also abandoned. Various gradient methods were investigated. The amount of overlap between blocks is explored. Different numbers of histogram bins, the effect of normalization methods, and block geometries are also explored. Although still numerous, these iterations of feature and parameter selection were more manageable than they could have been because of their preliminary selection of oriented gradients which was based on the success of SIFT features [9].

* Corresponding author.
E-mail addresses: kirt.lillywhite@gmail.com (K. Lillywhite), beau.tippetts@byu.edu (B. Tippetts), djlee@byu.edu (D.J. Lee).

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.05.012

In this paper we present a novel method for feature construction called Evolution-COnstructed Features (ECO features) which removes the need to tune parameters by a human expert. Feature construction differs from feature selection and feature extraction, which are more common in the literature. The following definitions are taken from Motoda and Liu [10].

Feature selection is a process that chooses a subset of features from the original features so that the feature space is optimally reduced according to a certain criterion.

Feature extraction is a process that extracts a set of new features from the original features through some functional mapping.

Feature construction is a process that discovers missing information about the relationships between features and augments the space of features by inferring or creating additional features.

The major contribution of our ECO features algorithm is that it constructs features from raw pixels through evolution searching for good features to describe the target of interest, taking a step towards solving the difficult problem of mapping signal to symbol. While good features could mean different things in different context, here it refers to features that provides information about an object that when used in conjunction with a machine learning algorithm are highly discriminative. The human expert is removed from the process of constructing the features for object recognition and is replaced by a genetic algorithm that automatically constructs good features. The constructed features provide information that cannot be found with the raw pixels by themselves. The genetic algorithm allows for a large number of features to be tested and for non-intuitive features to be considered. The constructed features are then used by AdaBoost [11] for classifying objects. AdaBoost is a meta-algorithm that combines the results of multiple weak classifiers to produce a stronger classifier. The AdaBoost algorithm is explained in Section 2.3.

Using ECO features provides many benefits.

1. Good features can be discovered without the use of a human expert.
2. Non-intuitive features can be constructed that are not normally considered by human experts.
3. ECO features are not limited to certain image sources including data originating from CMOS Sensors, synthetic aperture radar (SAR), infrared (IR), and potentially others such as magnetic resonance imaging (MRI), computed tomography (CT), X-ray, etc.
4. ECO features can be learned off-line for any object type. In other systems the human expert creates features that are good for one class of objects but may do poorly on other objects types.
5. ECO features can be global or local feature types [1] (Explained in Section 2.1).

There is a great deal of research on feature selection [12–16] and extraction [17–19] but a much smaller body on feature construction. In one of the more successful examples of feature selection, Dollár et al. reduce the dimensionality of an initial random set of features using Adaboost [5]. Each feature is taken from a single image transform where the set of image transforms includes: various color and grayscale transforms, Gabor filters, Difference of Gaussian, gradient histograms, and gradient magnitude. Although there are similarities between the work of Dollár et al. and this work, our main focus is on feature construction.

Feature construction has been used to improve general machine learning algorithms. Both [20,21] use genetic programming to construct features to improve the results of a decision tree. They build trees of operators that are applied to primitive features to construct richer features.

Feature construction has also been applied to object recognition using evolutionary techniques to build trees of mathematical operators and primitive features. Vafaie and De Jong [22] use genetic programming to construct features and then reduce the number and redundancy of their features through feature selection. Their trained features are then used by a decision tree to detect eyes. Brumby et al. [23] apply the same technique to find water in multi-spectral aerial-photography. Roberts and Claridge [24] use pixel statistics as their primitive features and then pull pasta shapes from a noisy background. Lin and Bhanu [25] use co-evolution to operate on SAR imagery to find tanks.

There is a large body of work that uses Adaboost or other boosting methods as part of object recognition. The more notable examples include object recognition using a cascade of classifiers [26], a joint boosting method that uses common features to do multi-class classification [27], non-rigid object detection [28], and action recognition from videos [29].

Presentation of our algorithm and results is organized as follows: The inspiration for and details of the algorithm are discussed in Section 2. We define ECO features (Section 2.1), show how they are constructed (Section 2.2), show how to train Adaboost (Section 2.3), and then finally how ECO features are used with Adaboost to classify objects (Section 2.4). In Section 3, we show that this algorithm generalizes well across several datasets, finding ECO features that effectively describe each of the objects in their respective datasets. Comparisons are given on multiple Caltech image datasets (Section 3.1), the INRIA Person dataset (Section 3.2), and a SAR image dataset, "Volcanoes on Venus" (Section 3.3). Discussion of the performance is given in Section 4. We conclude with a summary of this work and outline of future efforts in Section 5.

2. Algorithm

Much of what is done in object recognition aims to achieve what the human vision system is capable of. This tends to cause many to model systems and algorithms after various aspects of the human vision system. This work, to some extent, has also been patterned after human biological structures. The visual cortex of the human brain is made up of hierarchical layers of neurons with feedforward and feedback connections that grow and evolve from birth as the vision system develops. These connections throughout the visual cortex layers make up stages of human vision processing. While later stages tend to contain most of the feedback paths, connections from early stages of the hierarchy are mainly feedforward. These early stages of processing in the visual cortex are sensitive to visual stimuli such as lines of certain lengths and angles [30], spatial frequencies, and colors [31]. Serre et al. [32] suggest that basic image transforms such as derivative-of-Gaussian and Gabor filters model these early processing stages of the visual cortex. We too focus on modeling the early stages of processing by implementing basic image transforms connected in a feedforward manner. In a similar way that feedforward neural connections between these visual cortex layers are created and evolved in humans, our genetic algorithm evolves parameters of transforms and the feedforward processing paths they make up. These processing paths of transforms become the features in our algorithm that describe objects.

2.1. What is an ECO feature?

An ECO feature, as defined in Eq. (1), is a series of image transforms, where the transforms and associated parameters are determined by a genetic algorithm. In Eq. (1), V is the ECO feature output vector, n is the number of transforms the feature is composed of, T_i is the transformation at step i, V_i is the intermediate value at step i, ϕ_i is the transformation parameter vector at step i, and $I(x_1,y_1,x_2,y_2)$ is a subregion of the original image, I, indicated by the pixel range x_1, y_1, x_2, y_2.

$$V = T_n(V_{n-1},\phi_n)$$

$$V_{n-1} = T_{n-1}(V_{n-2},\phi_{n-1})$$

$$\vdots$$

$$V_1 = T_1(I(x_1,y_1,x_2,y_2),\phi_1) \tag{1}$$

Any transformation is possible but we are mostly interested in those transforms that can be found in a typical image processing library. Table 1 lists the set of image transforms used, ψ, along with the number of parameters associated with that transform. The values that these parameters can take on for a given transform T_i make up the set ξ_{T_i}. The number of transforms used to initially create an ECO feature, n, currently varies from 2 to 8 transforms. With the datasets that were used in testing it was found that the average ECO feature contained 3.7 transforms with a standard deviation of 1.7 transforms. The range 2–8 transforms allowed the search to focus where it was more likely to find results. We wanted ECO features to be able to be long and complicated if it yielded good results but found through experimentation that longer ECO features were less likely to yield good results. Fig. 1 shows graphical examples of two ECO features.

The transformations of a feature are applied to a subregion of the image which can range from a 1×1 pixel area to the whole image. Examples of subregions are shown in Fig. 2. Rather than making any assumptions about what the salient regions of the image are and defining a criteria for their selection, the genetic algorithm is used to search for the subregion parameters x_1, y_1, x_2, y_2. In this way the saliency of a subregion is not determined by the subregion itself, but in its ability, after being operated on by the transforms, to help classify objects. Subregions allow both global and local information to be captured. Local features are those features located at a single point or small region of an image, whereas global features cover a large region or the entire image [1]. The use of subregions allows each ECO feature to specialize at identifying different aspects of the target object.

Table 1
A list of image transforms available for the genetic algorithm to compose ECO features and the number of parameters the genetic algorithm must set for each transform.

Image transform	$\lvert\phi\rvert$	Image transform	$\lvert\phi\rvert$
Gabor filter	6	Sobel operator	4
Morphological erode	1	Morphological dilate	1
Gaussian blur	1	Adaptive thresholding	3
Histogram	1	Hough lines	2
Hough circles	2	Harris corner strength	3
Normalize	3	Histogram equalization	0
Discrete Fourier transform	1	Log	0
Square root	0	Median blur	1
Canny edge	4	Distance transform	2
Integral image	1	Laplacian edge detection	1
Difference of Gaussians	2	Rank transform	0
Census transform	0	Convert	0

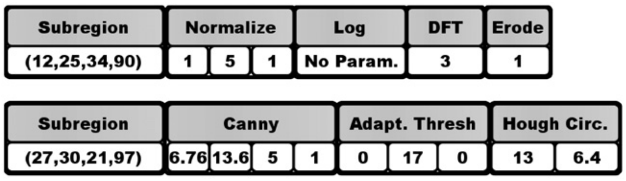

Fig. 1. Two example ECO features. The first example shows an ECO feature where the transforms are applied to the subregion where $x_1=12$, $y_1=25$, $x_2=34$, and $y_2=90$ from Eq. (1). The values below the transforms are the parameter vectors ϕ_i also from Eq. (1).

2.2. Constructing ECO features

ECO features are constructed using a standard genetic algorithm (GA) [33]. GAs, in general, are used for optimization and searching large spaces efficiently. They start with a population of creatures, representing possible solutions, which then undergo simulated evolution. Each creature is made up of genes which are the parameters of that particular solution. A fitness score, which is designed specifically for the problem, is computed for each creature and indicates how good the solution is. At each generation, creatures are probabilistically selected from the population to continue on to the next generation. Creatures with higher fitness scores are more likely to be selected. Other creatures are made through crossover, which combines the genes of two creatures to form one. Finally the genes of each creature in the population can possibly be mutated according to a mutation rate, which effectively creates a slightly different solution. The algorithm then ends at some predefined number of generations or when some criteria is satisfied.

In our algorithm, GA creatures represent ECO features. Genes are the elements of an ECO feature which include the subregion (x_1, y_1, x_2, y_2), the transforms $(T_1, T_2,\ldots,T_n)$, and the parameters for each transform ϕ_i. The number of genes that makeup a creature is not of fixed length since the number of transforms can vary and each transform has a different number of parameters. Initially, the genetic algorithm randomly generates a population of ECO features and verifies that each ECO feature consists of a valid ordering of transforms.

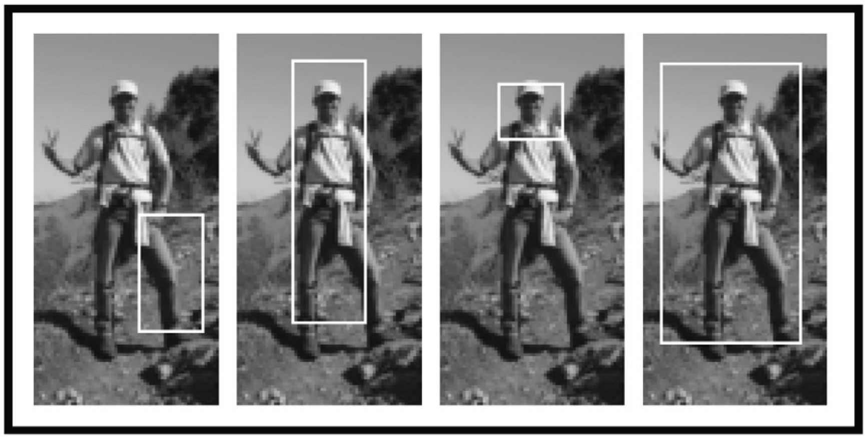

Fig. 2. Examples of subregions selected by the genetic algorithm.

In order to assign a fitness score to each ECO feature a weak classifier is associated with it. A single perceptron is used as the weak classifier as defined in Eq. (2). The perceptron maps the ECO feature input vector V to a binary classification, α, through a weight vector W and a bias term b.

$$\alpha = \begin{cases} 1 & \text{if } W \cdot V + b > 0 \\ 0 & \text{else} \end{cases} \tag{2}$$

Training the perceptron generates the weight vector W according to Eq. (3). Training images are processed according to Eq. (1) and the output vector V is the input to the perceptron. The error, δ, is found by subtracting the perceptron output, α, from the actual image classification β. The perceptron weights are updated according to this error and a learning rate λ.

$$\delta = \beta - \alpha$$

$$W[i] = W[i] + \lambda \cdot \delta \cdot V[i] \tag{3}$$

A fitness score, s, is computed using Eq. (4), which reflects how well the perceptron classifies a holding set. In Eq. (4), p is a penalty, t_p is the number of true positives, f_n is the number of false negatives, t_n is the number of true negatives, and f_p is the number of false positives. The fitness score is an integer in the range [0, 1000].

$$s = \frac{t_p \cdot 500}{f_n + t_p} + \frac{t_n \cdot 500}{p \cdot f_p + t_n} \tag{4}$$

Eq. (4) has two nice properties. Unlike classification accuracy, this fitness is not sensitive to unbalanced numbers of negative and positive training examples. For instance, if there are far more negative examples in the training set, a fitness score based on classification accuracy would favor a weak classifier that classifies everything as negative, although it has no ability to discriminate positive examples from negative examples. Eq. (4) also allows us to add a penalty p to false positives in order to bias the weak classifiers towards those with low false positive rates. Whenever a creature's fitness score is over a threshold it is added to a pool from which AdaBoost draws candidates.

After a fitness score has been obtained for every creature, a portion of the population is selected to continue to the next generation. A tournament selection method is used to select which creatures move to the next generation. A tournament selector takes N creatures at random and the creature with the best fitness score continues to the next generation. By adjusting N the ability for creatures with lower fitness scores to move to the next generation can be tuned. Higher values of N prohibit creatures with low fitness scores to move on to the next generation. Currently N is set to 2 which allows weaker creatures to move on. After selection has taken place the rest of the population is composed of new creatures created through crossover as shown in Fig. 3. Through the process of crossover it is possible for ECO features to have a transform length, n, longer than 8 which is the cap placed on gene length when they are being created. Once the next generation is filled, each of the parameters in the creatures can be mutated, also shown in Fig. 3. This whole process of finding features is summarized in Algorithm 1.

Algorithm 1. Finding Features

```
for Size of population do
  Randomly create creature
    $x_1 \in [0, \text{image width}-2]$,   $y_1 \in [0, \text{image height}-2]$,
  $x_2 \in [x_1+1, \text{image width}-1]$,   $y_2 \in [y_1+1, \text{image height}-1]$,
      $T_1(\phi_1) \in [\psi], \ldots, T_n(\phi_n) \in [\psi]$,
      $\phi_1 \in [\xi_{T_1}], \ldots, \phi_n \in [\xi_{T_n}]$
end for
for number of generations do
  for every creature do
    for every training image do
      Process image with feature transformations
      Train creature's perceptron
    end for
    for every holding set image do
      Process image with feature transformations
      Use perceptron output to update fitness score
    end for
    Assign fitness score to the creature
    Save creature if fitness score $>$ threshold
  end for
  Select creatures that make it to next generation
  Create new creatures using cross over
  Apply mutations to the population
end for
```

2.3. Training AdaBoost

After the genetic algorithm has completed finding good ECO features, Adaboost is used to combine the weak classifiers to make a stronger classifier. Algorithm 2 outlines how AdaBoost is trained.

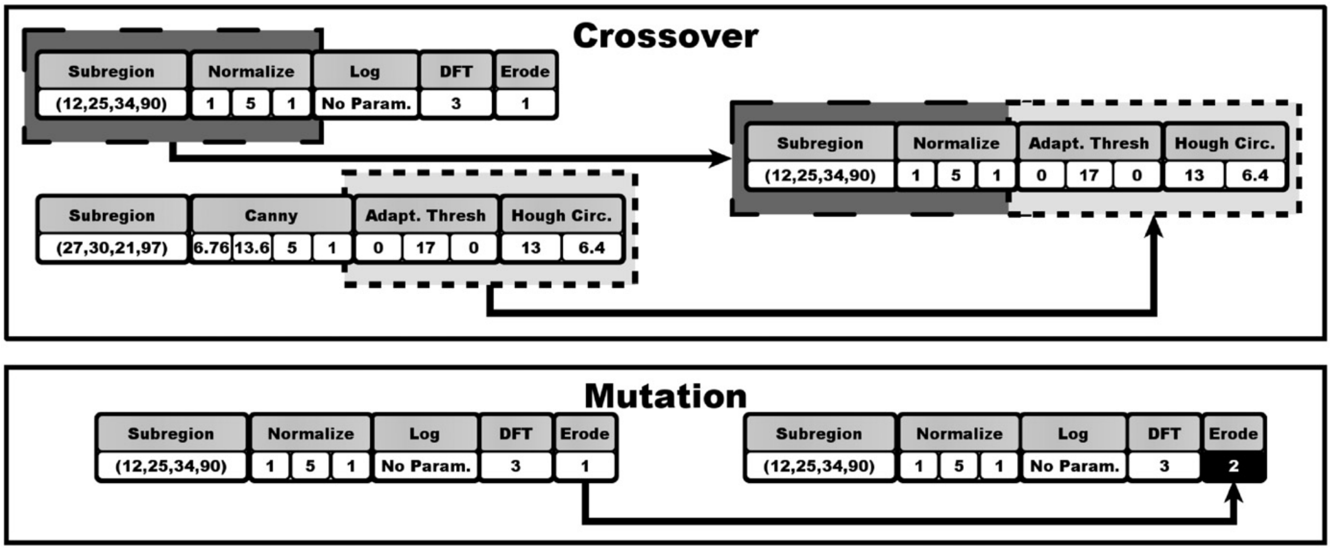

Fig. 3. Examples of crossover and mutation.

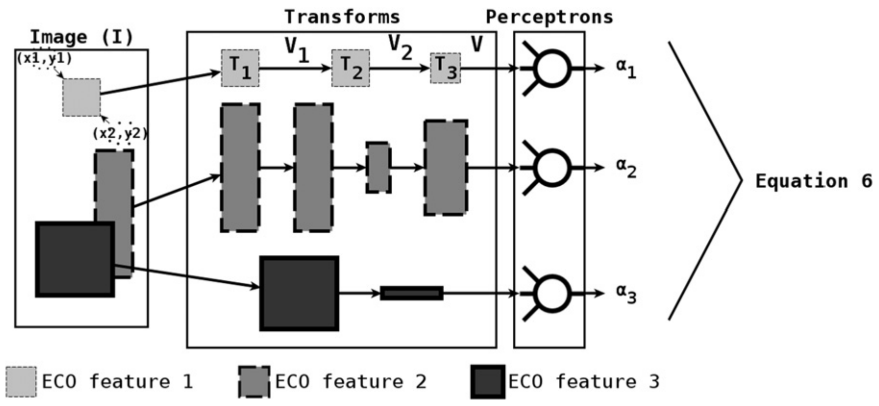

Fig. 4. ECO features and their corresponding perceptrons are combined using Adaboost to classify an image as object or non-object.

X represents the maximum number of weak classifiers allowed in the final model, which in our case is 400. X is set to 400 because on all of the datasets that were classified, none of them produced better results using more than 100 weak classifiers and the assumption was made that 400 would be sufficient for future datasets. The number of weak classifiers in the model can be reduced after training if desired. The normalization factor in Algorithm 2 is set so that the sum of the error over all the training images is 1. After training, the resulting AdaBoost model consists of a list of perceptrons and coefficients that indicate how much to trust each perceptron. The coefficient for each perceptron, ρ, is calculated using Eq. (5) where δ_w is the error of the perceptron over the training images. See Algorithm 2 to see more details about how the error is calculated.

When ECO features are created they are not allowed to have more than eight transforms, but through crossover it is possible for an ECO features to grow longer. In all of the experiments, ECO features consisting of more than eight transforms never performed well enough to be selected by Adaboost.

Algorithm 2. *Train AdaBoost*

```
Set of training images $M$
**for** every training image, $m$ **do**
  Initialize $\delta_M[m] = 1/|M|$
**end for**
**for** $x = 0$ to $X$ **do**
  **for** every perceptron, $w$, **do**
    **for** every training image, $m$ **do**
      **if** wrongly classified **then**
        $\delta_w += \delta_M[m]$
      **end if**
    **end for**
  **end for**
  Select perceptron with minimum error, $\Omega$
  **if** $\delta_\omega[\Omega] >= 50\%$ **then**
    BREAK
  **end if**
  Calculate coefficient of perceptron using Eq. (5)
  **for** every training image, $m$ **do**
  $c = \begin{cases} 1 & \text{if classified correctly by } \Omega \\ -1 & \text{else} \end{cases}$
  $\delta_M[m] = \frac{\delta_M[m] * e^{-\rho \cdot c}}{\text{Normalization Factor}}$
  **end for**
**end for**
```

$$\rho = \frac{1}{2} \cdot ln \frac{1-\delta_w}{\delta_w} \tag{5}$$

2.4. Using the AdaBoost model

Fig. 4 shows an example of classifying an image with an Adaboost model containing three ECO features. The figure shows each feature operating on its own subregion of the image (see Eq. (1)). It is possible for the subregions of different features to overlap. Also as the subregions pass through the transforms, the intermediate results may vary in size from one to the next. Each feature is accompanied by its trained perceptron. The output of each perceptron is combined according to Eq. (6) where X is the number of perceptrons in the Adaboost model, ρ_x is the coefficient for the perceptron x (see Eq. (5)), α_x is the output of perceptron x (see Eq. (2)), τ is a threshold value, and c is the final classification given by the Adaboost model. The threshold τ can be changed to vary the tradeoff between false positives and false negatives.

$$c = \begin{cases} 1 & \text{if } \sum_{x=1}^{X} \rho_x \cdot \alpha_x > \tau \\ 0 & \text{else} \end{cases} \tag{6}$$

2.5. Algorithm parameters

One of the main accomplishments of the ECO features algorithm is that a human expert is not needed for the construction of features or the tuning of their parameters. The initial low level features of the algorithm must be chosen but the algorithm is able to construct richer features from the low level features. The only inputs to the algorithm are the training images and the outputs are the self-tuned ECO features and the AdaBoost model used for object recognition. All the parameters of the transforms are tuned by the genetic algorithm.

Although the ECO features are self-tuned there are still parameters of the genetic algorithm that can be varied such as population size, the number of generations, mutation rate. Koppen et al. state that there is a large range of parameters that work when applying genetic algorithms to object recognition tasks [34]. Various experiments were performed to find the parameters of the genetic algorithm used here. Although the experiments were far from exhaustive, the following parameters were used to achieve the results discussed in the next section, a population size of 1000, evolved over 100 hundred generations, and using a mutation rate of 0.05.

3. Results

To test the algorithm, a variety of datasets were used. Again no parameters were tuned for any dataset, each was trained and tested in the exact same way. When a dataset was large enough to allow it, bootstrapping was done as part of training Adaboost.

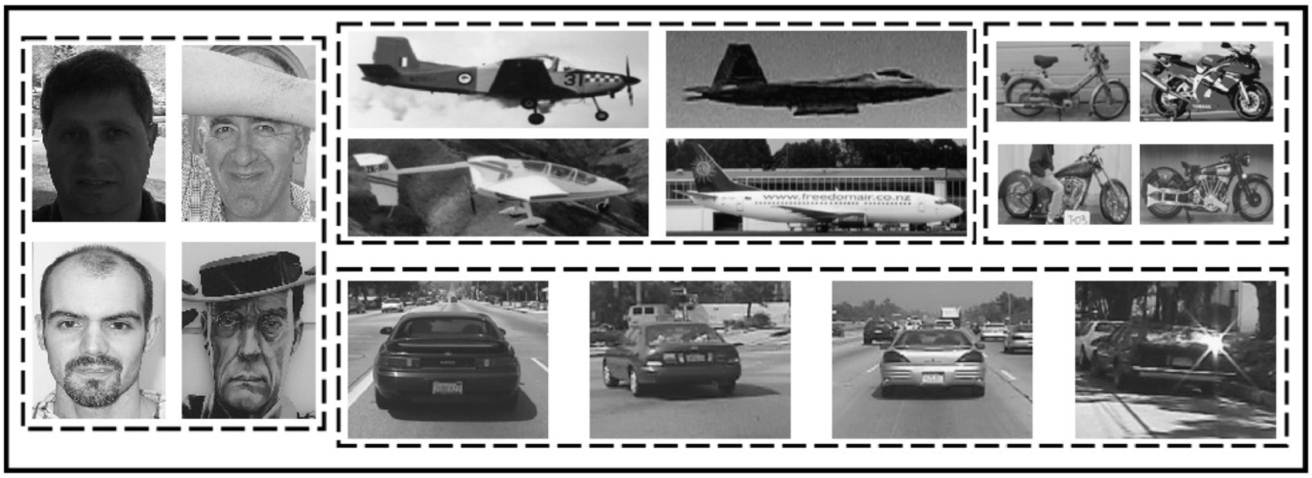

Fig. 5. Example images of the Caltech datasets after being scaled and color removed.

Bootstrapping is the process of retraining the Adaboost model on difficult examples that were previously misclassified [35].

For each dataset tested the best published results for that dataset were used for comparison. These methods are described below.

Fergus [36]: Fergus et al. create a constellation of parts and then represent the shape, appearance, occlusion, and relative scale using a probabilistic representation. Classification is done using a Bayesian approach. Their features look for salient regions over both location and scale.

Serre [37]: Serre et al. detect objects by building a hierarchical system that alternates between simple and complex layers. The simple layers combine their inputs with a bell-shaped tuning function and the complex layers combine their inputs through a maximum operation. The first layer consists a battery of Gabor filters at various scales and orientations.

Schwartz [6]: Schwartz et al. augment HOG features with texture and color. They perform feature extraction using partial least squares analysis to reduce the dimensionality of their feature set.

Tuzel [7]: Tuzel et al. use covariance matrices as object descriptors that are represented as a connected Riemannian manifold. Their main contribution is a novel method for classifying points that lie on a connected Riemannian manifold using the geometry of the space.

Burl [38]: Burl et al. identify volcanoes using a two pass system, one phase to identify candidate regions and another phase to do the final classification. They use PCA to reduce the input dimensionality and then use a Bayesian classifier.

3.1. Caltech datasets

Four datasets are used from the Caltech datasets [39]; motorcycles, faces, airplanes, and cars. Samples from the datasets are shown in Fig. 5. For each dataset,[1] the images were split into training and test sets according to [36] so that good comparisons could be made to other methods. Table 2 outlines the number of images in each dataset.

Table 3 shows the performance of ECO features, perfectly classifying all four Caltech datasets, alongside the results of other methods. Fergus et al. [36] and Serre et al. [37] represent the current best published results on the Caltech datasets. Schwartz et al. [6] did not publish results on the Caltech dataset but we tested their method on the Caltech datasets which was also able to perfectly classify the four datasets.

Table 2
Breakdown of the number of images in the Caltech datasets.

Dataset	Training images	Testing images
Motorbikes	400	400
Faces	218	217
Airplanes	400	400
Cars	400	400

Table 3
Comparison on Caltech datasets to other methods.

Dataset	Method			
	Fergus [36]	Serre [37]	Schwartz [6]	Ours
Motorbikes	95.0	98.0	100.0	100.0
Faces	96.4	98.2	100.0	100.0
Airplanes	94.0	96.7	100.0	100.0
Cars	95.0	98.0	100.0	100.0

Additional experiments were performed to see how many ECO features were actually required to still perfectly classify the datasets. Fig. 6 shows the results on the Caltech datasets using fewer ECO features. The axes in Fig. 6 are labeled with the standard metrics of precision and recall as defined in Eqs. (7) and (8) where t_p is the number of true positives, t_n is the number of true negatives, f_p is the number of false positives, and f_n is the number of false negatives. The number of ECO features used is reduced by lowering the parameter X used by Adaboost. X starts at 400 ECO features and then one by one the least influential ECO feature was removed. At most 25 features were needed to perfectly classify the faces and cars datasets while the airplanes dataset only needed ten features and the motorcycle dataset only needed four features to obtain perfect results. These results show that reducing the number of features, even drastically, still lead to good accuracy overall.

$$precision = \frac{t_p}{t_p+f_p} \tag{7}$$

$$recall = \frac{t_p}{t_p+f_n} \tag{8}$$

[1] http://www.robots.ox.ac.uk/~vgg/data3.html

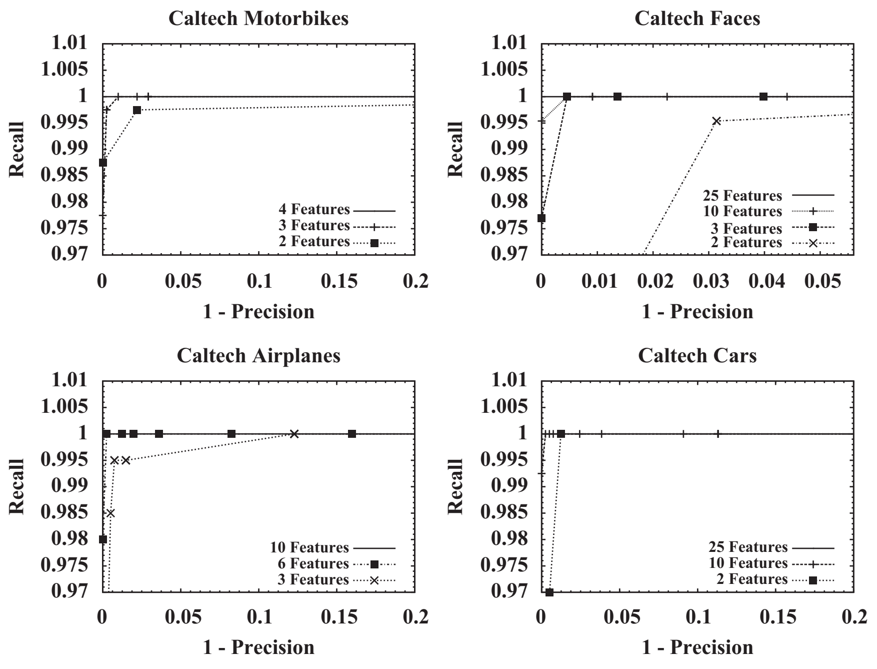

Fig. 6. Performance on the Caltech datasets using varying number of ECO features.

Fig. 7. Example images from the INRIA Person dataset.

Table 4
Summary of how the INRIA Person dataset was used. The label Pos stands for positive human examples and the label Neg stands for negative human example.

Training set				Testing set	
Find features		Train AdaBoost		Test algorithm	
Pos	Neg	Pos	Neg	Pos	Neg
1489	1576	2416	11,128	1132	20,356

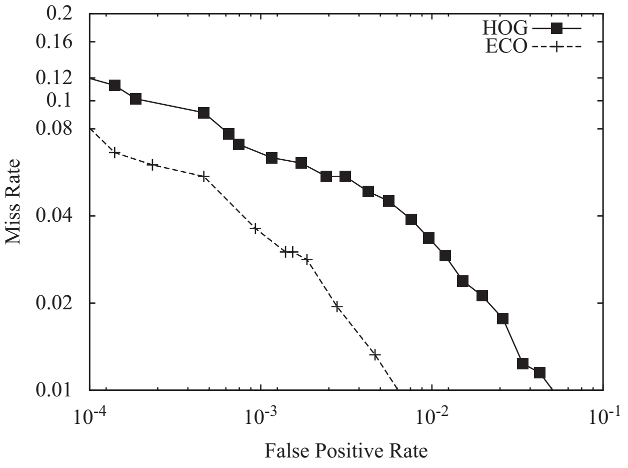

Fig. 8. Comparison to the HOG detection systems.

3.2. INRIA Person dataset

ECO features are tested using the INRIA Person dataset [40]. Examples of the dataset are shown in Fig. 7. The dataset is very challenging because it contains humans with large variations in pose, lighting conditions, background, clothing, and partial occlusion. Table 4 gives a break down of how the dataset was used.

The accuracy of ECO features alongside the very successful HOG method is shown in Fig. 8. What is exciting is that a human expert did not design these features, nevertheless the genetic algorithm was able to find something that was intelligent and effective, performing better than a feature set hand crafted by human experts.

The example ECO features shown back in Fig. 1 are some of the features constructed for human detection. A human expert would never had made the first feature that consisted of a normalize, followed by a log, a DFT, and then an erode operator. It is unclear what information this provides but the results show that it helps distinguish a human from a non-human.

Table 5 gives a summary of results for the state-of-the-art classification algorithms on the INRIA Person dataset. A false positive rate of 10^{-4} is a common comparison point for algorithms on this dataset. While ECO features perform better that the original HOG method it does not do as well when compared to newer methods. It should be noted, however, that there are no published methods that do better than HOG without including HOG as part of their method. For practicality purposes only basic image transformations have been currently implemented for use by ECO features. Making more image transforms available to ECO features could help take advantage of information it currently cannot capture, e.g. oriented gradients, color information, etc. See Section 5 for plans of future implementations.

Fig. 9 shows the results when varying the number of ECO features used. Again we started with 400 ECO features and then one by one removed the least influential ECO feature. Since 100 or more ECO features gives almost identical results, only 100 ECO features is shown in order to make the graph more readable. Reducing the number of features by 90% gives only a 7% reduction in recall at a 90% precision rate.

Table 5
Comparison to state-of-the-art methods on the INRIA Person dataset.

Method	Miss rate at 10^{-4} false positive rate (%)
HOG	12
Ours	8
Dollár [5]	7
Tuzel [7]	7
Dollár [41]	4
Schwartz [6]	3

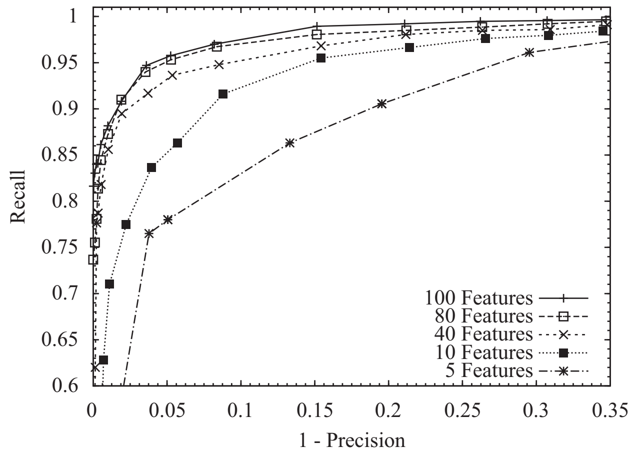

Fig. 9. Performance of ECO features on the INRIA Person dataset using varying number of features.

3.3. Volcanoes on Venus

The volcanoes of Venus dataset [42] is one result of the Magellan mission to Venus from 1989 to 1994 and contains 134 1024 × 1024 SAR images. With the volume of images resulting from the mission, automated methods for analyzing the data were sought out [38].

The volcanoes on Venus dataset is a very challenging dataset to classify even for humans. The top row in Fig. 10 shows examples of volcanoes and the bottom row contains sample images without a volcano. Although the images represent the highest SAR resolution available at the time, they still lack detail to make the volcanoes obviously distinguishable from the rest of the planet. In an effort to quantify uncertainty, examples were labeled by geologists into five categories: category 1—almost certainly a volcano, category 2—probably a volcano, category 3—possibly a volcano, category 4—a visible pit but not much evidence of other volcano traits, category 5—no volcano. This labeling represents the ground truth

Table 6
Confusion matrix created by two expert geologist and together represent the ground truth for the dataset.

Geologist A	Geologist B				
	Label 1	Label 2	Label 3	Label 4	Label 5
Label 1	19	9	13	1	4
Label 2	8	8	12	4	8
Label 3	4	6	18	5	29
Label 4	1	5	1	24	16
Label 5	3	5	37	15	X

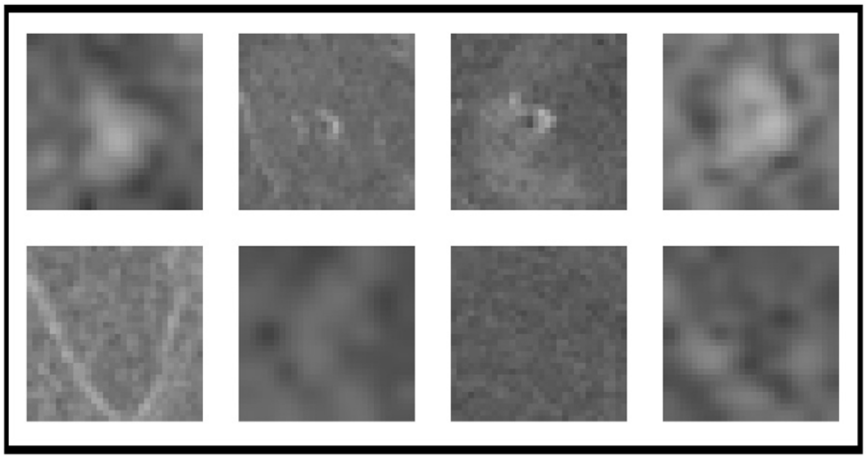

Fig. 10. Example images from the volcanoes on Venus dataset. The first row are positive examples and the second row are negative examples.

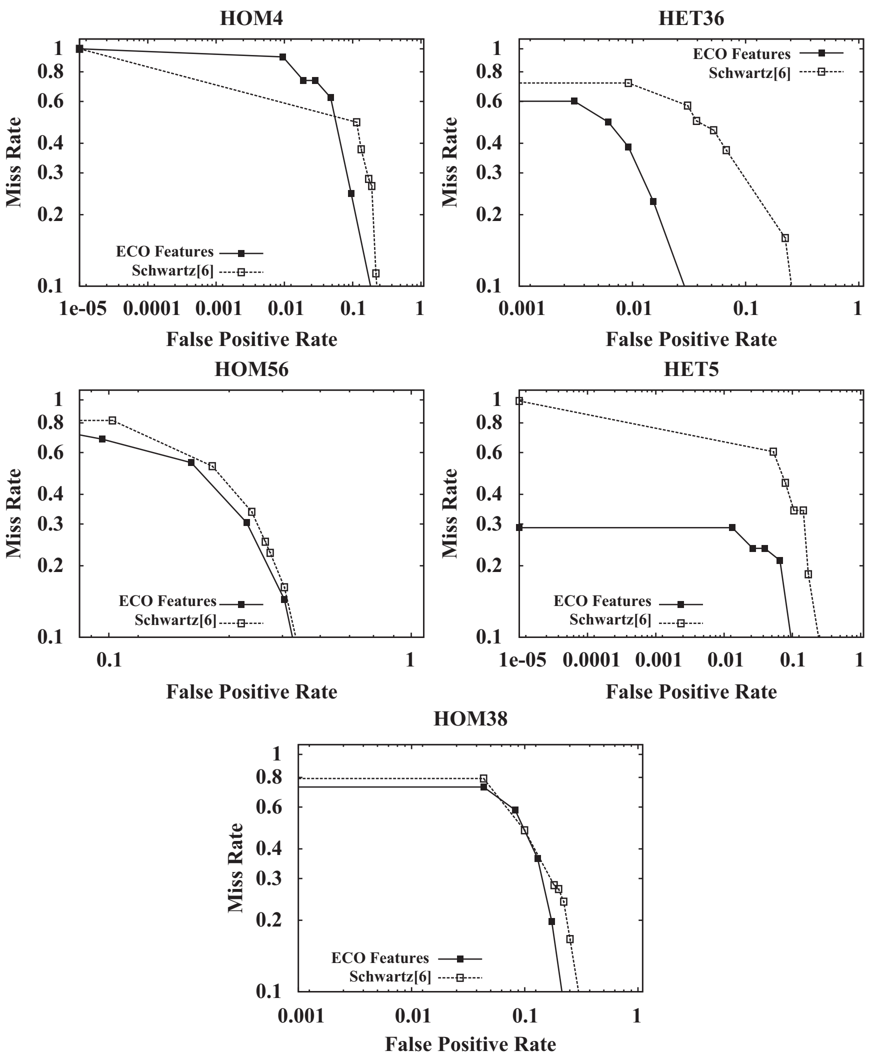

Fig. 11. Comparison on the volcanoes on Venus dataset, partition HET36, to the Schwartz [6] method.

for the dataset. A confusion matrix of labeling given by two geologists is shown in Table 6 and indicates how difficult the dataset is, given that the ground truth is so difficult to establish. A confusion matrix compares the classification of two entities, generally the predicted classification and the actual classification. Here the labeling of one geologist is compared against the other geologist. If they agreed on every volcano then the confusion matrix, for example, would show that whenever geologist A labeled the volcano as category 1 geologist B would also label it as category 1.

ECO features were trained using multiple 30×30 pixel regions from each image. The dataset is partitioned into five parts called HOM4, HOM38, HOM56, HET5, and HET36. HOM means that the images are fairly homogeneous and HET means that the images were taken from different parts of the planet and have more variability, and the number indicates the number of images in the partition. Within each partition, cross-validation is performed rotating which images are used for testing. Half of the volcano examples taken from the ground truth were used for training and the other half for testing. An equal number of non-volcano samples were sampled randomly from the rest of the image, taking care that there was no overlap with labeled volcanoes. When training the AdaBoost models not all 400 weak classifiers were included because the error rates for the weak classifiers reached 50% (see Algorithm 2).

An attempt was made to do a comparison to Burl et al. [38] but there was insufficient information to make a fair and accurate comparison. A comparison could be made against the Schwartz method [6] by downloading available code.[2] Their method was trained and tested using the same image sets as was used for ECO features. The results of this test are shown in Fig. 11. ECO features outperformed Schwartz on every test and did significantly better on the HET partitions. It appears that the Schwartz method [6] performs better on the HOM4 partition for miss rates greater than 50%, but this is only a result of the low number of images in the partition. Removing the last point in the graph gives a better view of the results, since both have 100% miss rate at zero false positives.

4. Commentary on performance

When one stops to think about the number of image transform combinations, their parameters, and the possible locations on the image to apply the transforms—the search space seems dizzyingly large. Surprisingly, however, for all experiments run so far, good ECO features have been found very quickly. Even after a few generations of the genetic algorithm, weak classifiers with good discriminative power are found. To help with run times, the image transform functions from OpenCV are used, which are optimized for fast performance.

[2] http://www.umiacs.umd.edu/~schwartz/softwares.html

Most of the training was done on an AMD Phenom II 920 quad-core running at 2.8 GHz, with four gigabytes of RAM. The Caltech datasets took approximately 20 min each to train. The volcanoes on Venus dataset, which consists of 20 different training sets, took approximately 2 h to train altogether. The INRIA Person dataset consisted of a far greater number of images and trained in about 2 h using a cluster of three Dell PowerEdge M610 with two Xeon Quad-core X5560 processors. Although these training times are not critical, as training is intended to be an off-line process, they demonstrate that training time is not exorbitant nor impractical. It should be noted that as training images are added, or as the number of generations or size of population in the genetic algorithm increase, the training times scale linearly.

To take advantage of multiple cores, the inner loop of Algorithm 1, where over 99% of the computation is done, was parallelized using OpenMP. OpenMP is a library that provides multi-threading with the use of compiler directives and library routines. OpenMP has a construct for parallelizing for loops, and in this situation, one simple compiler directive was all that was needed to take advantage of all the processors and cores on our machines.

5. Conclusion and future work

It has been shown that ECO features generalize well across datasets based only on a set of basic image transforms. Results show that ECO features have high discriminative properties for target objects and that the performance on each dataset was either superior or comparable to state-of-the-art methods. Given the variability that exists between datasets, however, we believe that ECO features can be further expanded and improved with the addition of other image transforms. For example, we plan to add HOG, a transform which is used in other object recognition methods, including top performing human recognition algorithms [41,6,7,5,43]. Color in some datasets can provide a good deal of information and would also be a valuable addition to the set of image transforms ECO features could choose from. In the future, plans include analyzing the image transforms and the contributions of individual ECO features.

Perhaps the most exciting result of this work is that the human expert has been removed from the loop of constructing features. There are no parameters to tune or tweak for a given dataset and object. All that is needed is a training set and ECO features take care of the rest, making object detection accessible to almost any user. We hope that this will be a powerful tool for the scientific community to use, and will allow other researchers to employ object detection as part of their work without being experts in feature construction. There is no need for a human expert to trial and error different features, drastically reducing development time.

The fact that image transforms are applied to a subregion of the image, allows ECO features to be viewed as a part-based object recognition algorithm. Part-based methods generally do better under occlusions [44]. While partial occlusions do appear in the datasets in our experiments, future work will include tests to analyze performance on occluded objects.

References

[1] P. Roth, M. Winter, Survey of Appearance-Based Methods for Object Recognition, Technical Report, Institute for Computer Graphics and Vision, Graz University of Technology, 2008.
[2] V. Cherkassky, Y. Ma, Practical selection of SVM parameters and noise estimation for SVM regression, Neural Networks, vol. 17, Elsevier, 2004, pp. 113–126.
[3] N. Cristianini, C. Campbell, J. Shawe-Taylor, Dynamically adapting kernels in support vector machines, in: Advances in Neural Information Processing Systems, Citeseer, 1999, pp. 204–210.
[4] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2005, pp. 886–893.
[5] P. Dollár, Z. Tu, P. Perona, S. Belongie, Integral channel features, in: British Machine Vision Conference, 2009.
[6] W. Schwartz, A. Kembhavi, D. Harwood, L. Davis, Human detection using partial least squares analysis, in: IEEE International Conference on Computer Vision, 2009, pp. 24–31.
[7] O. Tuzel, F. Porikli, P. Meer, Pedestrian detection via classification on riemannian manifolds, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2008) 1713–1727.
[8] P. Felzenszwalb, D. McAllester, D. Ramanan, A discriminatively trained, multiscale, deformable part model, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[9] D. Lowe, Object recognition from local scale-invariant features, The Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, 1999, pp. 1150–1157.
[10] H. Motoda, H. Liu, Feature selection, extraction and construction, Communication of Institute of Information and Computing Machinery, vol. 5, Citeseer, 2002, pp. 67–72.
[11] Y. Freund, R. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the Thirteenth International Conference on Machine Learning, 1996, pp. 148–156.
[12] P. Narendra, K. Fukunaga, A branch and bound algorithm for feature subset selection, IEEE Transactions on Computers C-26 (1977) 917–922.
[13] J. Bala, K. Jong, J. Huang, H. Vafaie, H. Wechsler, Using learning to facilitate the evolution of features for recognizing visual concepts, Evolutionary Computation 4 (1996) 297–311.
[14] A. Jain, D. Zongker, Feature selection: evaluation, application, and small sample performance, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (1997) 153–158.
[15] M. Dash, H. Liu, Consistency-based search in feature selection, Artificial Intelligence 151 (2003) 155–176.
[16] C. Huang, C. Wang, A GA-based feature selection and parameters optimization for support vector machines, Expert Systems with Applications 31 (2006) 231–240.
[17] D. Tao, X. Li, X. Wu, S. Maybank, General tensor discriminant analysis and gabor features for gait recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2007) 1700–1715.
[18] S. Yeom, B. Javidi, E. Watson, Three-dimensional distortion-tolerant object recognition using photon-counting integral imaging, Optics Express 15 (2007) 1513–1533.
[19] T. Kim, J. Kittler, Locally linear discriminant analysis for multimodally distributed classes for face recognition with a single model image, IEEE Transactions on Pattern Analysis and Machine Intelligence (2005) 318–327.
[20] M. Smith, L. Bull, Genetic programming with a genetic algorithm for feature construction and selection, Genetic Programming and Evolvable Machines 6 (2005) 265–281.
[21] K. Neshatian, M. Zhang, M. Johnston, Feature construction and dimension reduction using genetic programming, Lecture Notes in Computer Science, vol. 4830, Springer, 2007, pp. 160–171.
[22] H. Vafaie, K. De Jong, Feature space transformation using genetic algorithms, IEEE Intelligent Systems and their Applications 13 (1998) 57–65.
[23] S. Brumby, J. Theiler, S. Perkins, N. Harvey, J. Szymanski, J. Bloch, M. Mitchell, Investigation of image feature extraction by a genetic algorithm, Proceedings of SPIE 3812 (1999) 24–31.
[24] M. Roberts, E. Claridge, Cooperative coevolution of image feature construction and object detection, in: Lecture Notes in Computer Science, Springer, 2004, pp. 902–911.
[25] Y. Lin, B. Bhanu, Learning features for object recognition, in: Lecture Notes in Computer Science, Springer, 2003, pp. 2227–2239.
[26] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, IEEE Computer Society, Los Alamitos, CA, USA, 2001, pp. 511–518.
[27] A. Torralba, K.P. Murphy, W.T. Freeman, Sharing features: efficient boosting procedures for multiclass object detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, IEEE Computer Society, Los Alamitos, CA, USA, 2004, pp. 762–769.
[28] M. Andriluka, S. Roth, B. Schiele, Pictorial structures revisited: people detection and articulated pose estimation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1014–1021.
[29] J. Liu, J. Luo, M. Shah, Recognizing realistic actions from videos “in the wild”, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1996–2003.
[30] B.G. Maddox, C.L. Swadley, An Intelligent Systems Approach to Automated Object Recognition: A Preliminary Study. Open-file Report. U.S. Geological Survey, U.S. Department of the Interior, 2002.
[31] J. Jones, L. Palmer, An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex, Journal of Neurophysiology 58 (1987) 1233–1259.
[32] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, T. Poggio, A theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex, in: AI Memo, 2005, pp. 1–131.
[33] M. Mitchell, An Introduction to Genetic Algorithms, The MIT Press, 1998.
[34] M. Koppen, K. Franke, R. Vicente-Garcia, Tiny gas for image processing applications, IEEE Computational Intelligence Magazine 1 (2006) 17–26.

[35] E. Osuna, R. Freund, F. Girosi, et al., Training support vector machines: an application to face detection, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 24, 1997, pp. 130–136.
[36] R. Fergus, P. Perona, A. Zisserman, Object class recognition by unsupervised scale-invariant learning, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, 2003, pp. 264–271.
[37] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, Robust object recognition with cortex-like mechanisms, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2007) 411–426.
[38] M. Burl, L. Asker, P. Smyth, U. Fayyad, P. Perona, L. Crumpler, J. Aubele, Learning to recognize volcanoes on Venus, Machine Learning 30 (1998) 165–194.
[39] Caltech Computation Vision Group, Caltech Image Datasets. 〈http://www.vision.caltech.edu/html-files/archive.html〉.
[40] National Institute for Research in Computer Science and Control, INRIA Person Dataset. 〈http://pascal.inrialpes.fr/data/human/〉, 2005.
[41] P. Dollár, B. Babenko, S. Belongie, P. Perona, Z. Tu, Multiple component learning for object detection, in: European Conference on Computer Vision, 2008, pp. 211–224.
[42] A. Asuncion, D. Newman, UCI Machine Learning Repository, 〈http://www.ics.uci.edu/~mlearn/MLRepository.html〉, 2007.
[43] S. Maji, A. Berg, J. Malik, Classification using intersection kernel support vector machines is efficient, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[44] K. Mikolajczyk, C. Schmid, A. Zisserman, Human detection based on a probabilistic assembly of robust part detectors, in: Proceedings of the Eighth European Conference on Computer Vision, Prague, Czech Republic, 2004, pp. 69–82.

Kirt Lillywhite received his B.S. degree in computer engineering from Brigham Young University, Provo, Utah, in 2006. He is currently pursuing a Ph.D. degree from the Department of Electrical and Computer Engineering at Brigham Young University. His research interests include real-time image processing, pattern recognition, parallel processing, and robotic vision.

Beau Tippetts received his M.S. and B.S. from Brigham Young University, Provo, Utah, USA. His research interests include real-time vision algorithm optimization using FPGAs, in part for use on hovering micro-UAVs. Specifically areas include feature detection, tracking, stereo, obstacle avoidance. He is currently a Ph.D. student at Brigham Young University. He is a member of IEEE.

Dah-Jye Lee received his B.S. degree from National Taiwan University of Science and Technology in 1984, M.S. and Ph.D. degrees in electrical engineering from Texas Tech University in 1987 and 1990, respectively. He also received his MBA degree from Shenandoah University, Winchester, Virginia in 1999. He is currently a Professor in the Department of Electrical and Computer Engineering at Brigham Young University. He worked in the machine vision industry for eleven years prior to joining BYU in 2001. His research work focuses on Medical informatics and imaging, shape-based pattern recognition, hardware implementation of real-time 3-D vision algorithms and machine vision applications. Dr. Lee is a senior member of IEEE.