Pattern Recognition 46 (2013) 3300–3314

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

# A feature construction method for general object recognition

Kirt Lillywhite, Dah-Jye Lee *, Beau Tippetts, James Archibald

*Department of Electrical and Computer Engineering, Brigham Young University, Provo, Utah 84601, USA*

ARTICLE INFO

*Article history:*
Received 19 September 2012
Received in revised form
1 April 2013
Accepted 2 June 2013
Available online 15 June 2013

*Keywords:*
Object detection
Genetic algorithm
Feature construction
ECO features
AdaBoost

ABSTRACT

This paper presents a novel approach for object detection using a feature construction method called Evolution-COnstructed (ECO) features. Most other object recognition approaches rely on human experts to construct features. ECO features are automatically constructed by uniquely employing a standard genetic algorithm to discover 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, and no limitations to certain types of image sources. We show in our experiments that ECO features perform better or comparable with hand-crafted state-of-the-art object recognition algorithms. An analysis is given of ECO features which includes a visualization of ECO features and improvements made to the algorithm.



## 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 and articulation, distortions, high intraclass variation, image sensor noise, wide variability in the types of cues necessary to detect the specific object type, partial occlusions, 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 which include automotive safety, surveillance, video indexing, image classification, content-based image retrieval, tracking, robotics, and a host of others. Despite the large number of applications that would benefit from robust object recognition and other incentives, object recognition is still not yet widely adopted in industry.

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. In object recognition these symbols are referred to as features. Machine learning techniques are commonly used to take these features and then classify them as either belonging to the object of interest or not. In general, machine learning algorithms take in symbols, find patterns in the symbols, and use mathematical methods to separate the symbols into classes. Machine learning frees the user from having to identify rules for classification and in general is more accurate at creating rules than human experts are. Machine learning techniques, however, are most successful when the set of features uniquely describes the object of interest. The image processing used to create a higher-level representation of the input signal bridges the semantic gap that exists between the raw input signal and what is needed by the machine learning algorithm. Agarwal and Roth state it this way "…we suggest that in order to extract high-level, conceptual information such as the presence of an object in an image, it is essential to transform the raw, low-level input (in this case, the pixel grayscale values) to a higher-level, more 'meaningful' representation that can support the detection process [2]."

The various methods that have been used to obtain high quality features can be categorized into three groups: feature selection, feature extraction, and feature construction. The following definitions are taken from Motoda and Liu [3]:

*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.

Fig. 1 shows where feature selection, extraction, and construction fit into the process of object recognition. Feature construction will generate a set of symbols from the raw data that is obtained

* Corresponding author. Tel.: +1 801 422 5923; fax: +1 801 422 0201.
*E-mail addresses:* kirt.lillywhite@gmail.com (K. Lillywhite), djlee@byu.edu (D.-J. Lee), beau.tippetts@byu.edu (B. Tippetts), jka@byu.edu (J. Archibald).

0031-3203/$ - see front matter © 2013 Published by Elsevier Ltd.
http://dx.doi.org/10.1016/j.patcog.2013.06.002


**Fig. 1.** The steps of object recognition with the forms that the data takes during the process.

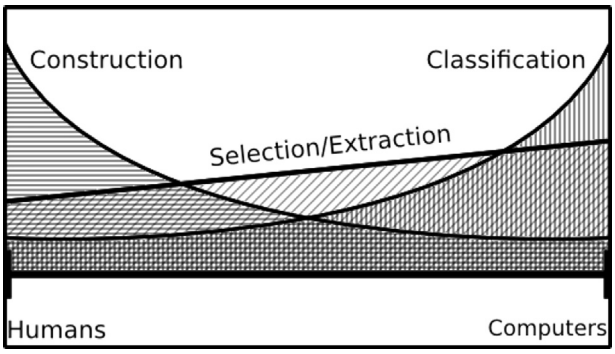


**Fig. 2.** Shows a current look at what tasks are performed by whom.

from the imaging sensor. Those symbols are enhanced by choosing a subset of the most important symbols through feature selection, or creating new symbols through some functional mapping by performing feature extraction. Then the symbols are classified as belonging to the object or not.

Feature construction is mostly performed by humans currently, while the classification step is almost always performed by computers. Feature extraction and selection on the other hand have been performed by both but with the scales tipping more toward computers. Fig. 2 shows a current estimated distribution of which object recognition steps is performed humans versus computers. In the object recognition literature, the pattern of human experts creating features and then using machine learning algorithms for classification is clear [2,4–16].

In the same way that the application of machine learning has improved the accuracy of object recognition, we believe that feature construction can have the same kind of impact on improving the quality of features. And as stated before, higher quality features are more able to uniquely describe the object of interest, producing more accurate object recognition. One major reason for this is the ability of a computer to find patterns in large amounts of data that humans simply cannot do. In the same way that machine learning frees the user from having to generate its own rules for classification, feature construction frees the user from having to generate their own features.

The main focus of this work is to produce a high quality feature construction algorithm capable of general object recognition. Rather than relying on human experts our method, which we call Evolution-COnstructed (ECO) features, uses simulated evolution to construct series of transforms that convert the input signal of raw pixels into high quality features [17–19]. Using ECO features also 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.

## 2. Background

As the amount of data increases and the desire to capture all the information possible, dimensionality reduction through feature selection is becoming much more important. There is a good deal of literature on feature selection [20,21]. Narendra and Fukunaga use a branch and bound method to avoid an exhaustive search of all feature subsets but still select the best subset according to their criterion [22]. Liu et al. also develop a feature selection method to avoid exhaustive search for a bag-of-visual words applications [23]. Dash and Liu evaluate the effectiveness of various feature selection methods using an inconsistency measure and explore various search strategies for feature selection. Wang et al. use particle swarm optimization to do feature selection in a stochastic way [24]. Pedrycz and Ahmad do feature selection, again in a stochastic manner, using both a genetic algorithm and particle swarm optimization but using structure retention as their criteria [25]. These methods highlight a progression from exhaustive methods, to search based methods that find the best subset possible, to stochastic methods to deal with the size of the feature selection space.

Feature selection is also used for a few other applications. Huang et al. use feature selection to find a subset of features for a support vector machine [26]. Bala et al. use a genetic algorithm and decision trees to select features for visual concepts [27]. Dollár et al. reduce the dimensionality of an initial random set of features using AdaBoost [28] in order to detect humans in images. Sun et al. do feature selection by using a genetic algorithm to select a subset of eigenvectors rather than the traditional method of selecting some percentage of the top eigenvectors, and then test their method on vehicle and face detection [29].

Feature extraction is most commonly seen as a way to reduce the dimensionality of the feature space by using a function that finds linear combinations of the most important features. Linear discriminate analysis and principle component analysis are very common methods for feature extraction [30,31]. Variations of linear discriminate analysis have also been done for feature extraction [32]. Sherrah et al. use genetic programming to decide whether to do feature selection or extraction [33] as a pre-processing step before classification.

Feature construction methods appear much less frequently in the literature than feature selection and feature extraction methods. Feature construction has been used to improve general machine learning algorithms. Both [34] and [35] 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 [36] 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. [37] apply the same technique to find water in multi-spectral aerial-photography. Roberts and Claridge [38] use pixel statistics as their primitive features and then pull pasta shapes from a noisy background. Bulitko et al. do automatic image interpretation that search for transforms that allow remote sensing images to be automatically segmented [39]. Krawiec and Bhanu use series of transforms found through evolution to

construct features [40]. Although somewhat similar to the ECO feature algorithm, in the end they use a set of feature extraction operators (various moments and other pixel statistics) to construct a real-valued feature vector of predetermined length. This technique forces a specific dimensionality and constrains the features to be of a certain class, namely those that can be generated from the set of extraction operators selected by the authors.

None of the feature construction techniques that we could find in the literature, and discussed here, performs well at general object detection. Generally they are tested on a single dataset with various constraints. The authors believe that the real power of a feature construction algorithm is manifest when it is able to automatically generate sufficiently unique features to perform well at general object recognition.

Automatic feature construction is also related to deep learning. Both are aimed at automatically finding a better representation of the image. Deep learning methods have had good success on many datasets. Cirşan et al. use a deep six layer neural network to do handwritten digit recognition [41]. Their method is currently the best known method on the MNIST handwritten digits dataset achieving a 0.23% error rate on the dataset. Krizhevsky et al. achieve top-1 and top-5 error rates of 37.5% and 17.0% respectively [42] in the ImageNet LSVRC-2010 contest. Both these methods use deep neural networks optimized to run on GPUs and use the processing power of the GPU to overcome the "vanishing gradients problem" that exists in deep neural networks. Le et al. use deep learning methods of stacking and convolution to learn hierarchical representations of images [43]. Their work uses unlabeled video data to find features like other unsupervised feature learning methods such as Sparse Coding and Stacked Autoencoders.

There are many works on general object recognition. Viola and Jones, in a very well known work, developed an algorithm using AdaBoost to select basic integral image features and build a cascade of detectors so that background images are quickly classified. Mohan et al. built a parts based detector with Haar wavelets to represent the image. They then use a support vector machine for classification [44]. Belongie et al. create a shape descriptor that represents shape with a set of discrete points sampled from the contour of the shape and then use $K$ nearest neighbors for classification [5]. Agarwal and Roth use Förstner operator to find representative parts of the target object and create image patches that can then be used to represent the object [2]. While by itself it is not an object recognition algorithm, Lowe developed an image feature he called SIFT, that became the basis for features in many object recognition algorithm [45]. Schneiderman and Kanade built a classifier based on the statistics of localized parts. Each part is described using a subset of wavelet coefficients [11]. Dalal and Triggs created a feature called histograms of oriented gradients that excelled at many object detection tasks [46] and was modeled in part from SIFT features. Many works since then use some form of histograms of oriented gradients [4,47,7,8,13]. Laptev uses histograms as features, weighted Fisher linear discriminant as a weak classifier, and then AdaBoost for classification. Serre et al. Bileshi, and Siagian and Itti developed biologically inspired features that are based on Gabor filters [48,6,49].

## 3. ECO feature algorithm

ECO features are constructed using a genetic algorithm which creates an ordering of basic image transforms. The set of transforms that are available to the genetic algorithm are shown in Table 1. According to the definition of an ECO feature, Eq. (1), the ECO feature output vector, $V$, is created by applying each of $n$ transforms to a subregion, $I(x_1, y_1, x_2, y_2)$, of an input image, $I$. Each

**Table 1**
A list of image transforms available to the genetic algorithm for composing 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 |
| Gradient | 1 | Difference of Gaussians | 2 |
| Square root | 0 | Morphological erode | 1 |
| Gaussian blur | 1 | Adaptive thresholding | 3 |
| Histogram | 1 | Hough lines | 2 |
| Hough circles | 2 | Fourier transform | 1 |
| Normalize | 3 | Histogram equalization | 0 |
| Convert | 0 | Laplacian edge | 1 |
| Median blur | 1 | Distance transform | 2 |
| Integral image | 1 | Morphological dilate | 1 |
| Canny edge | 4 | Harris corner strength | 3 |
| Rank transform | 0 | Census transform | 0 |
| Resize | 1 | Pixel statistics | 2 |
| Log | 0 | | |

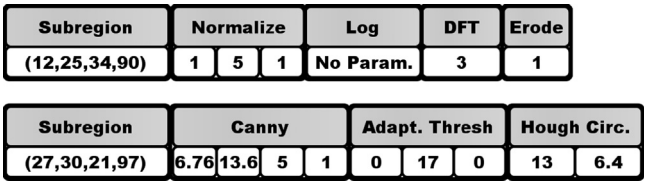


**Fig. 3.** Two example ECO features. The first example shows an ECO feature where the transforms are applied to the subregion where $x_1 = 12$, $y_1 = 25$, $x_2 = 34$, and $y_2 = 90$ from Eq. (1). The values below the transforms are the parameter vectors $\phi_i$, also from Eq. (1).

transform, $T_i$, of the series is applied to the output of the previous transform, $V_{i-1}$, using the transform parameters in vector $\phi_i$, which are also set by the genetic algorithm. Fig. 3 shows graphical examples of two ECO features

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

$$V_{n-1} = T_{n-1}(V_{n-2}, \phi_{n-1})$$
$$\vdots$$
$$V_1 = T_1(I(x_1, y_1, x_2, y_2), \phi_1)$$

In addition to generating a sequence of transforms and their operating parameters, the genetic algorithm also selects which portion of the image the transforms will operate on. Any amount of the image may be selected as the input to the transform series, $I(x_1, y_1, x_2, y_2)$, from the whole image down to a subregion as small as a single pixel. Examples of subregions are shown in Fig. 4, which shows how the selection of the subregion causes ECO features to specialize at an aspect of the target object. 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 automatically 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. The use of subregions allows each ECO feature to specialize at identifying different aspects of the target object.

### 3.1. Constructing ECO features

ECO features are constructed using a standard genetic algorithm (GA) [50]. GAs, in general, are used for optimization and

**Fig. 4.** Examples of subregions selected by the genetic algorithm.

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. Some new 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. This process is repeated for several generations, evolving solutions so that they have better fitness scores. The algorithm ends at some predefined number of generations or when some criteria for fitness scores 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 $\phi_i$. The number of genes that make up 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.

In order to assign a fitness score to each ECO feature a weak classifier is associated with it. The fitness score is related to how well this classifier identifies an object in a small set of training images. 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, $\alpha$, 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, $\delta$, is found by subtracting the perceptron output, $\alpha$, from the actual image classification $\beta$. The perceptron weights are updated according to this error and a learning rate $\lambda$

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

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]

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

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.

**Algorithm 1.** *Finding features.*

```
for Size of population do
  Randomly create creature.
  Select x1, y1, x2, y2, T1(φ1), ..., Tn(φn).
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
```

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. After selection has taken

place, the rest of the population is composed of new creatures created through crossover as shown in Fig. 5. 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. 5. This whole process of finding features is summarized in Algorithm 1.

### 3.2. Training AdaBoost

After the genetic algorithm has found good ECO features, AdaBoost is used to combine the weak classifiers, associated with each ECO feature, to make a stronger classifier. Algorithm 2 outlines how AdaBoost is trained. $X$ represents the maximum number of weak classifiers allowed in the final model. 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, $\rho$, is calculated using Eq. (5) where $\delta_w$ is the error of the perceptron over the training images. See Algorithm 2 for more details about how the error is calculated. The AdaBoost method is slightly different than many implementations that use perceptrons as the weak classifier. Generally the perceptron is created according to the error distribution of the examples. ECO features are generated before AdaBoost is trained and hence there is a finite set of pregenerated perceptrons that can be selected for use.

**Algorithm 2.** *Train AdaBoost.*

Set of training images $M$
**for** every training image, $m$ **do**
  Initialize $\delta_M[m] = 1 \quad /| \quad M|$
**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**

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

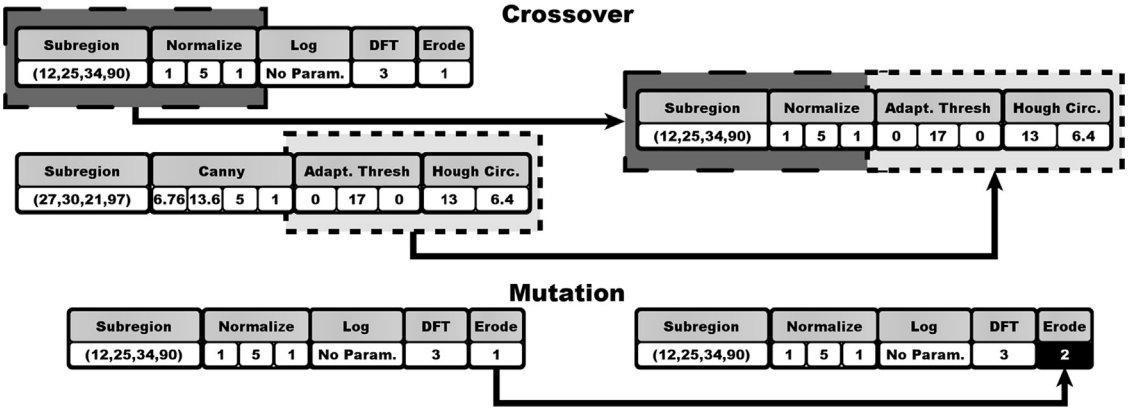


**Fig. 5.** Examples of crossover and mutation.

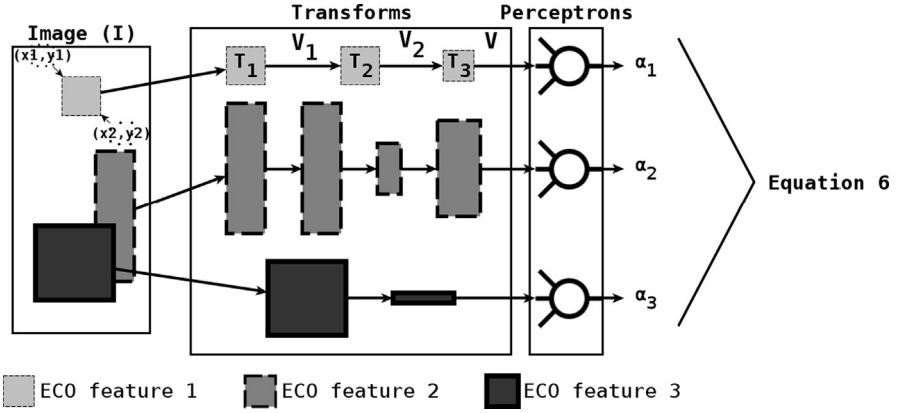


**Fig. 6.** ECO features and their corresponding perceptrons are combined using AdaBoost to classify an image as object or non-object.

**Fig. 7.** Example images of the Caltech datasets after being scaled and having color removed.

### 3.3. Using the AdaBoost model

Fig. 6 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)). As can be seen, 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, $\rho_x$ is the coefficient for the perceptron $x$ (see Eq. (5)), $\alpha_x$ is the output of perceptron $x$ (see Eq. (2)), $\tau$ is a threshold value, and $c$ is the final classification given by the AdaBoost model. The threshold $\tau$ can be changed to vary the tradeoff between false positives and false negatives

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

## 4. ECO feature performance

To test the ECO features algorithm, a variety of datasets were used. No parameters were tuned for any dataset, each was trained and tested in the exact same way. For each dataset tested, the best published results for that dataset were used for comparison. These methods are described below:

**Fergus [51].** 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 [52].** 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 [53].** 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 [54].** 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.

**Norouzi [55].** Norouzi et al. use a deep learning method consisting of a stack of convolutional restricted Boltzmann machines.

**Burl [56].** 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.

**Table 2**
Comparison on Caltech datasets to other methods.

| | Method | | | |
|---|---|---|---|---|
| Database | Fergus [51] | Serre [52] | Schwartz [53] | ECO |
| 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 |

### 4.1. Caltech datasets

Four of the Caltech datasets [57] were used; motorcycles, faces, airplanes, and cars. Samples from the datasets are shown in Fig. 7. For each dataset,[1] the images were split into training and test sets according to [51] so that good comparisons could be made to other methods.

Table 2 shows the performance of ECO features, perfectly classifying all four Caltech datasets, alongside the results of other methods. Fergus et al. [51] and Serre et al. [52] represent the current best published results on the Caltech datasets. Schwartz et al. [53] 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.

### 4.2. INRIA person dataset

ECO features were also tested using the INRIA person dataset [58]. Examples of the dataset are shown in Fig. 8. The dataset is very challenging because it contains humans with large variations in pose, lighting conditions, background, clothing, and partial occlusion. It was created by Dalal and Triggs because the existing

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

**Fig. 8.** Example images from the INRIA person dataset.

person datasets were not challenging enough for their histograms of oriented gradients method [46].

Table 3 gives a summary of results for the state-of-the-art classification algorithms on the INRIA person dataset on a per-window miss rate. A false positive rate of $10^{-4}$ is a common comparison point for algorithms on this dataset. ECO features perform just as well as the state-of-the-art on the INRIA person dataset.

**Table 3**
Comparison to state-of-the-art methods on the INRIA person dataset.

| Method | Miss rate at $10^{-4}$ false positive rate (%) |
|---|---|
| HOG | 12 |
| Dollár [28] | 7 |
| Tuzel [54] | 7 |
| Norouzi [55] | 7 |
| Dollár [59] | 4 |
| Schwartz [53] | 3 |
| ECO features | 3 |

### 4.3. Volcanoes on Venus

The volcanoes of Venus dataset [60] is one result of the Magellan mission to Venus from 1989 through 1994 and contains 134 $1024 \times 1024$ SAR images. With the volume of images resulting from the mission, automated methods for analyzing the data were sought out [56]. The dataset is a challenging one, even for human experts, and samples can be seen in Fig. 9. The ground truth for the dataset is a labeling created by geologists.

ECO features were trained using multiple $30 \times 30$ 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.

An attempt was made to do a comparison to Burl et al. [56] but there was insufficient information to make a fair and accurate comparison. It was possible to make a comparison against the Schwartz method [53] 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. 10. ECO features outperformed Schwartz on every test and did significantly better on the HET partitions. Eqs. (7) and (8) define the miss rate and false positive rate

$$\text{miss rate} = \frac{fn}{tp + fn} \tag{7}$$

$$\text{false positive rate} = \frac{fp}{tp + fp + tn + fn} \tag{8}$$

### 4.4. BYU fish dataset

The fish images used are from field study images taken by our university's biology department. The fish were captured, photographed, and released. There are four fish species represented in the dataset: yellowstone cutthroat, cottid, speckled dace, and whitefish. Samples of each species from the dataset are shown in Fig. 11.

Five fold cross validation was performed to test the ability of ECO features to distinguish each fish species. Each image in the dataset of a chosen species was treated as the positive example and all the other species made up the negative examples. Using five fold cross validation one fold is treated as the test set and the remaining four folds are used for training the ECO features. Once the ECO features are found, the images in the current fold are used to compute a classification accuracy. The results are given in Table 4. The average classification accuracy is 99.4%, with a standard deviation of 0.64%.

## 5. Visualization of ECO features

As has been presented, the performance of ECO features on object detection is, on individual datasets, among the best, and generalizes well across multiple datasets. Here, the focus has been turned from the raw performance to taking a closer look at how ECO features are composed and what the genetic algorithm is finding.

Since ECO features are performing a series of image transforms, what the genetic algorithm is doing can, in many cases, be understood by analyzing the images produced after each

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

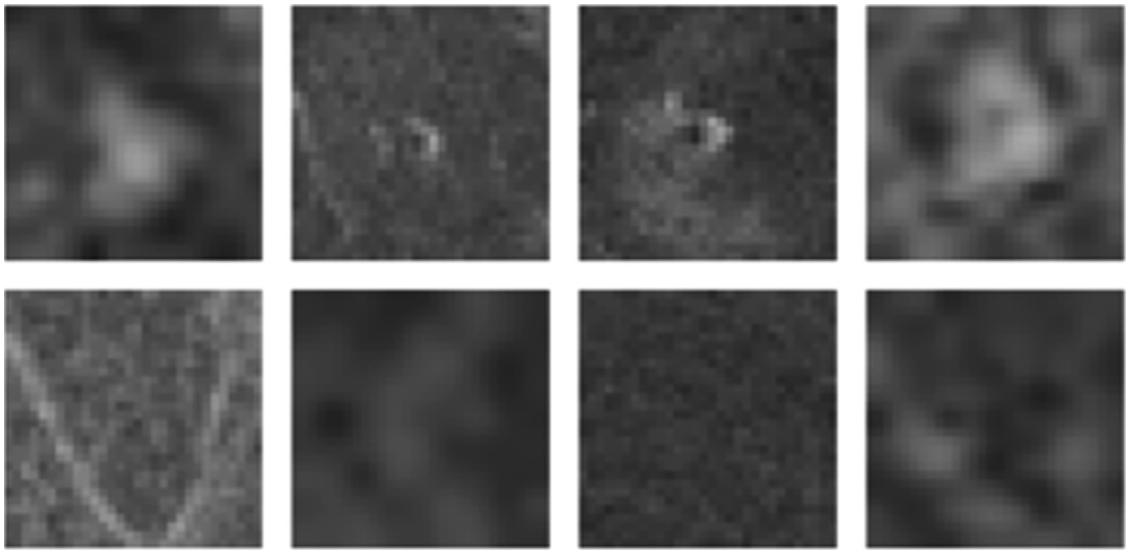

**Fig. 9.** Example images from the volcanoes on Venus dataset. The first row is positive examples of volcanoes and the second row is negative examples.

transform, $V_i$ (Eq. (1)). These images are produced by averaging all the positive or negative examples, after each transform, and then normalizing the result so it can be viewed as an image. Normalization is done according to Eq. (9). While the normalization is necessary to view the average output of the transforms as an image, it also has some negative effects. The normalization causes the perceived contrast differences to be relative and the actual image intensity magnitudes are not evident. The normalization of the average over the positive and negative examples is done separately which makes some comparisons between the two difficult. Despite the disadvantages the images do provide very clear information about what the genetic algorithm is finding

$$V_i(x,y) = \frac{V_i(x,y) - \min(V_i)}{\max(V_i) - \min(V_i)} \tag{9}$$

The output of the final transform, $V$, becomes the input to the perceptron of the ECO feature. Those inputs are multiplied by the perceptron weights, $W$. The greater the magnitude of the perceptron weight, the more important the input that is connected to that weight is. So if the perceptron weights are viewed as an image, in the same way that the output of the other transforms is viewed as an image, the relative importance of the inputs to the perceptron can be viewed. The positive and negative magnitude weights are viewed separately so that the importance of the weights for classifying the image as object or non-object can be seen. The visualizations give an understanding of what information the ECO features found.

Fig. 12 shows this visualization of ECO features on many of the datasets used in this work. Each letter represents an ECO feature and each number represents a transform in the order that they appear in the ECO feature. Images marked with an asterisk are averaged over the negative examples rather than the positive examples. The letters $n$ and $p$ at the end of the label indicate that the image is a representation of the weights of the perceptron, the negative weights $n$ and the positive weights $p$. The transforms represented in the figure are as follows: a1-Kirsh compass kernel in $x$ direction, a2-dilate, b1-Laplacian, b2-difference of Gaussians, b3-histogram equalization, c1-dilate, c2-Hough circles, d1-histogram equalization, d2-dilate, d3-Harris corner, e1-Gabor, e2-Kirsch compass gradient in the $y$ direction, e3-dilate, e4-Gabor, e5-median blur, f1-Gabor, f2-Gaussian Blur, g1-erode, g2-adaptive threshold, h1-erode, h2-dilate, h3-Hough Circles, i1-histogram equalization, i2-erode, j1-normalize, j2-Sobel, j3-dilate, k1-histogram equalization, k2-distance transform, k3-Sobel operator, l1-Harris corner, l2-integral image, m1-resize, m2-dilate, n1-difference of Gaussians, n2-simple gradient in $x$ direction, n3-Laplacian, n4-Sobel operator, n5-normalize, n6-census, n7-simple gradient in $x$ direction, n8-Gaussian b lur, o1-Gabor, o2-erode, p1-resize, p2-Gabor, q1-difference of Gaussians, q2-convert type, q3-resize, r1-difference of Gaussians, and r2-normalize.

Fig. 12 shows that shape information is the most common piece of information being extracted from the images. In ECO feature (a), the shape of a commercial plane stands out with the positive perceptrons having the greatest magnitude on the tail of the plane. There are two ECO features shown that feature a hough circle transform to find (c) the curve on the front of a plane and (h) the curve of a human chin. ECO feature (d) is discriminates part of the back of a vehicle from the road where there is a strong contrast difference. ECO feature (g) classifies as face, images with dark regions on the eyes and around the nose, and classifies as non-face, images with dark regions in other areas. ECO feature (i) and (j) look at the general shape of a motorbike. ECO feature (l) looks at the shape of a human head, ECO feature (m) the shape of a human at the waist, and (n) the general shape of a human with emphasis at the head and feet. ECO feature (o) looks at the position of the dorsal fin and adipose fin to identify Whitefish from the other three species of fish. ECO feature (p) is looking at the general shape of the Whitefish with emphasis on the head.

There are a few ECO features that are also looking at texture. The visualization of ECO feature (b) shows no discernible shape information. Fig. 13 shows the average of all the positive examples in the training set of the Caltech airplanes dataset. The subregion of ECO feature (b), however, is at the top of the image and as shown in the average airplane image, this appears to most often not to intersect with the airplane in the image. Many of the images in the Caltech airplanes dataset show the aircraft in the sky, where the background does not have much texture. The random images that make up the negative examples however tend to have a lot of texture and it seems this is what the ECO feature is trying to find. ECO feature (k) is fairly similar to (b) with a subregion that does not intersect the target object and instead focuses on the texture that exists mostly in the negative examples.

ECO feature (f) appears to be detecting the shadow of the vehicle which is much darker on average than the street scenes that make up the negative examples. Using a vehicle's shadow was

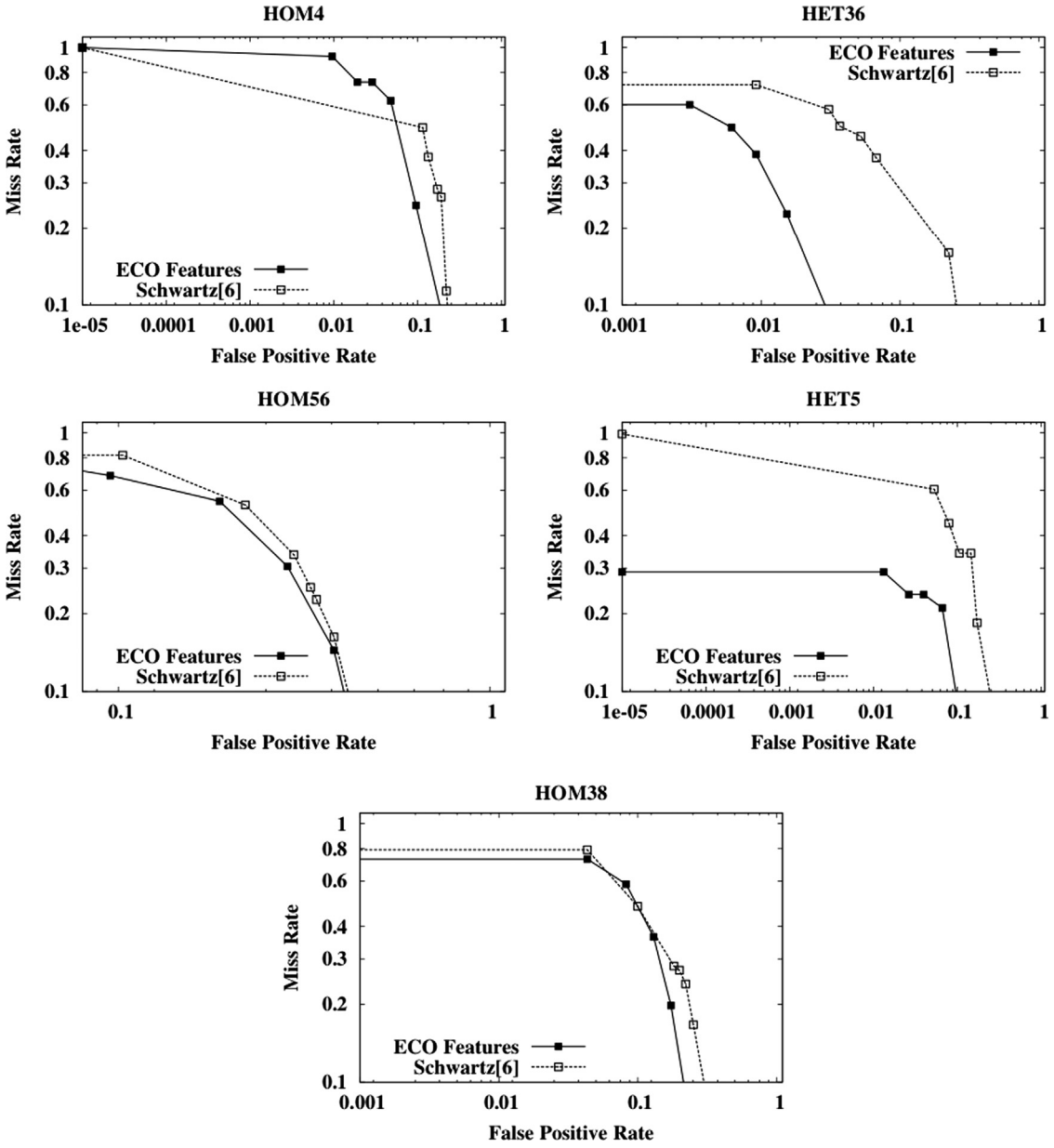


**Fig. 10.** Comparison on the volcanoes on Venus dataset, partition HET36, to the Schwartz [53] method.

recently proposed by Rosebrock et al. as a means to localize vehicles [61]. Without human intervention the ECO features algorithm is able to pick up on many of the same traits that human experts also target.

There are many ECO features that contain transforms that do not lend themselves to visualization such as a hough transforms like in (c3) and (h3), a distance transform shown in (k2), and integral images like (l2). Other transforms that are hard to visualize but not shown in Fig. 12 include histograms, discrete Fourier transform, and a pixels statistics transform. ECO feature (e), however, uses transforms that normally can be visualized but does not make clear in this case what information is being exploited.

The Caltech motorbikes and cars datasets use road images, with the absence of cars and motorbikes, as their negative examples. Using a specific set of negative images allows information to extracted from both the positive and negative examples. ECO feature (f2*) shows a bright road in the averaged image. ECO feature (i2*) and (j3*) even have the lines of the road displayed.

## 6. Understanding of ECO features

In addition to viewing what the genetic algorithm was finding and what information the ECO features were keying into, an analysis was made into the different parts of the ECO features algorithm.

### 6.1. Speciation

Genetic algorithms discard creatures with the lowest fitness scores in each generation, which eventually leads to the convergence of the entire population. In general, if the fitness truly reflects the strength of the solution, this is the desired behavior. There are situations, however, where certain types of solutions evolve slower than other types of solutions, but if given the opportunity could eventually evolve to have a similar or even better fitness score. For example, Fig. 14 shows an example search space. Searches starting at randomly chosen locations will most likely converge on solution A because of the gradient and size in

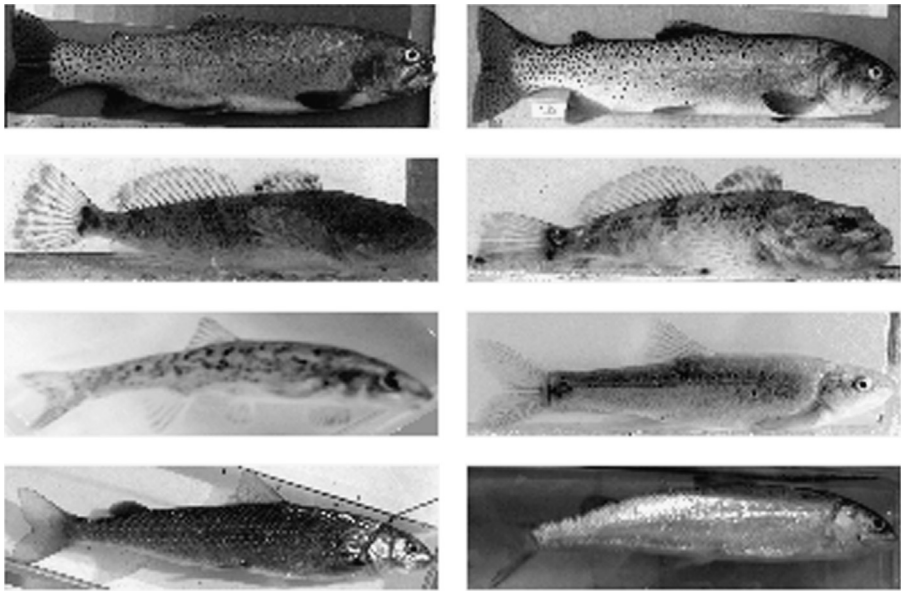

**Fig. 11.** Examples of the four fish species. From the top row to the bottom row the species are yellowstone cutthroat, cottid, speckled dace, and whitefish.

**Table 4**
The classification accuracy when doing five-fold cross validation (one fold for testing and four for training) for each species. One species is treated as the positive examples while the other species form the negative examples.

| Species | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Y. Cutthroat | 99.3 | 99.3 | 100 | 100 | 100 |
| Cottid | 100 | 100 | 100 | 99.3 | 98.4 |
| Speckled dace | 99.3 | 99.3 | 100 | 100 | 99.3 |
| Whitefish | 97.8 | 100 | 98.6 | 98.6 | 99.2 |

fitness score the hill around solution A provides. Searches that start off of the hill around point A could find point B, which has a slightly higher fitness score, but it is going to take longer and cannot be discarded too soon because of its low fitness score. Speciation preserves diversity and innovation by allowing creatures to compete in niches rather than competing against the entire population [62,63].

In order for speciation to occur, ECO features must be allowed to compete in niches, rather than against a large population. One way to allow speciation to occur would be to define species and only allow ECO features to compete if they were from the same species. With 27 possible transforms and millions of possible combinations of those transforms, defining species is very difficult. It is hard to determine a distance measure between sequences of transforms that does not divide the ECO features arbitrarily.

A simpler method to provide speciation, that does not alter the genetic algorithm, is to train ECO features in smaller populations. In large populations certain combinations of transforms were observed to appear frequently. If, however, those combinations of transforms were not present, other transform combinations over time could mature into solutions with equally high fitness scores. This observation shows that some combinations of transforms mature much faster within the genetic algorithm but do not necessarily perform any better than other combinations of transforms that mature more slowly.

In order to show the advantages of speciation, 500 ECO features were trained on the INRIA person dataset using a population size of 1000 and 500 ECO features were trained using a population size of 50. Fig. 16 shows the diversity of transforms and combinations of transforms between an AdaBoost model trained using speciation and without speciation. The circles in the figure represent the various possible transforms. The larger the circle the more frequently that transform appears in the AdaBoost model. Each transform has lines that connect it to transforms that follow it in the transform sequence. This highlights the diversity that speciation allows. The model with speciation has many more combinations of transforms that appear and the counts for each transform is more distributed.

In our previous paper we compared ECO features to the histogram of oriented gradients (HOG) method by Dalal and Triggs [46] on the INRIA person dataset [58]. Fig. 15 shows their comparison to the HOG method alongside our ECO features with added speciation. On the INRIA person dataset, speciation decreased the miss rate 5% points at $10^{-4}$ false positive rate, giving a 3% miss rate. ECO features with speciation is as good as or better than other state-of-the-art methods on the INRIA person dataset. Speciation allow ECO features to better compete with other state-of-the-art methods while still maintaining the ability of ECO features to adapt itself to different target objects.

### 6.2. ECO feature length

The number of transforms used to create an ECO feature, $n$, varies from 2 to 8 transforms. This decision was made to allow ECO features to be long and complicated if necessary but not so long that too much time was spent on complicated features that are unlikely to yield good results. To test the validity of this decision several AdaBoost models were trained on the INRIA person dataset where the range of allowable ECO feature lengths were varied. A comparison of the accuracy of these models is given in Fig. 17. Using only two transforms has a higher miss rate at all false positive rates and a significantly higher miss rate at low false positive rates. Including ECO features with three transforms gives a significant lower miss rate at low false positive rates. Adding more ECO features that are allowed to be longer helps a little until accuracy start to decrease slightly after allowing ECO features with more than seven transforms. An obvious advantage of using shorter ECO features is computation times. The testing times on the INRIA person dataset ranged from 5 to 9 min, on our workstation using a AMD Phenom 1055 T, depending on the range of lengths included in the model.

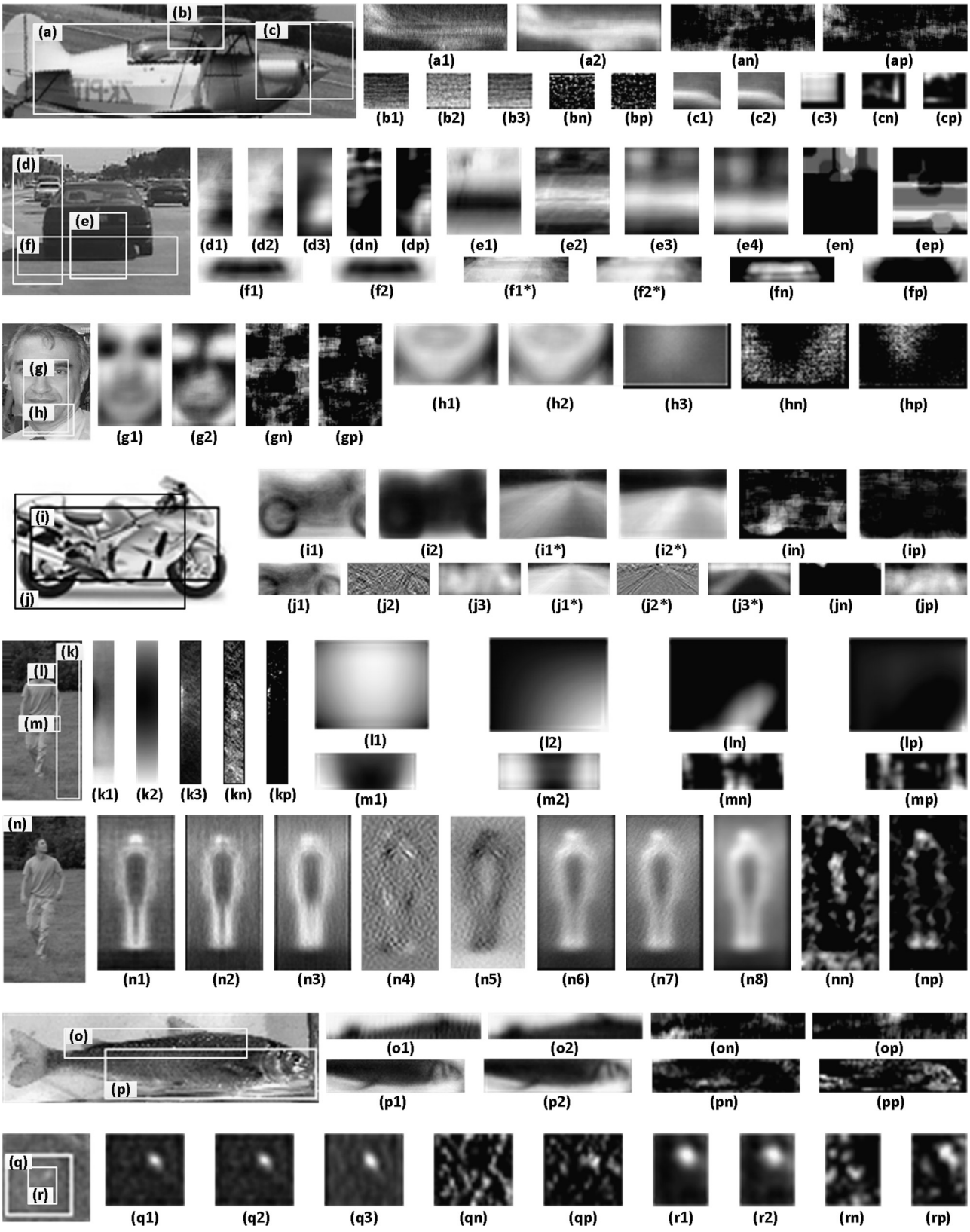


**Fig. 12.** Visualization of ECO features trained on many of the datasets tested in this work.

### 6.3. Subregions

ECO features were designed to use subregions to allow each ECO feature to specialize in identifying a different aspect of the target object. Further testing has shown that the use of subregions does not improve nor degrade the accuracy of ECO features. The use of subregions does, however, provide a 1.75 speedup testing on the INRIA person dataset. Subregions tend to be fairly large with

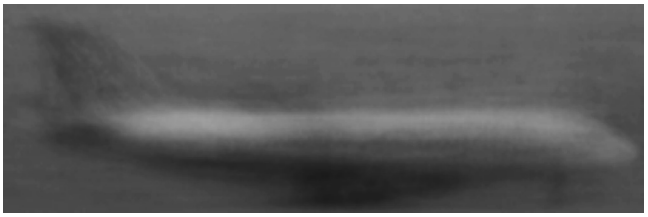

**Fig. 13.** The average over all the airplanes in the training set of the Caltech airplanes dataset.

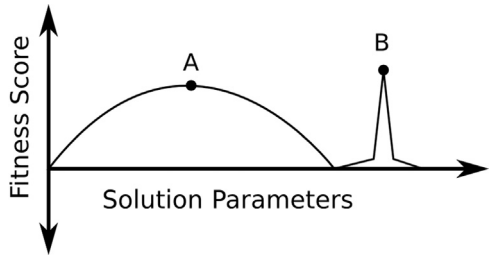


**Fig. 14.** A search algorithm to find the maximum fitness score is much more likely find point A before point B.

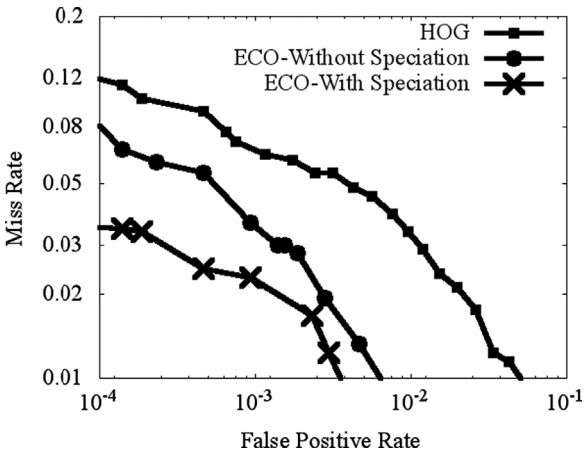


**Fig. 15.** Comparison between histogram of oriented gradients, ECO features, and ECO features with speciation on the INRIA person dataset.

an average area approximately equal to 35% of the entire image area. For example, over the INRIA person dataset where the full size images are $64 \times 128$, the average subregion size is $37 \times 79$. Fig. 18 shows the comparison on the INRIA person dataset with subregions turned on and off.

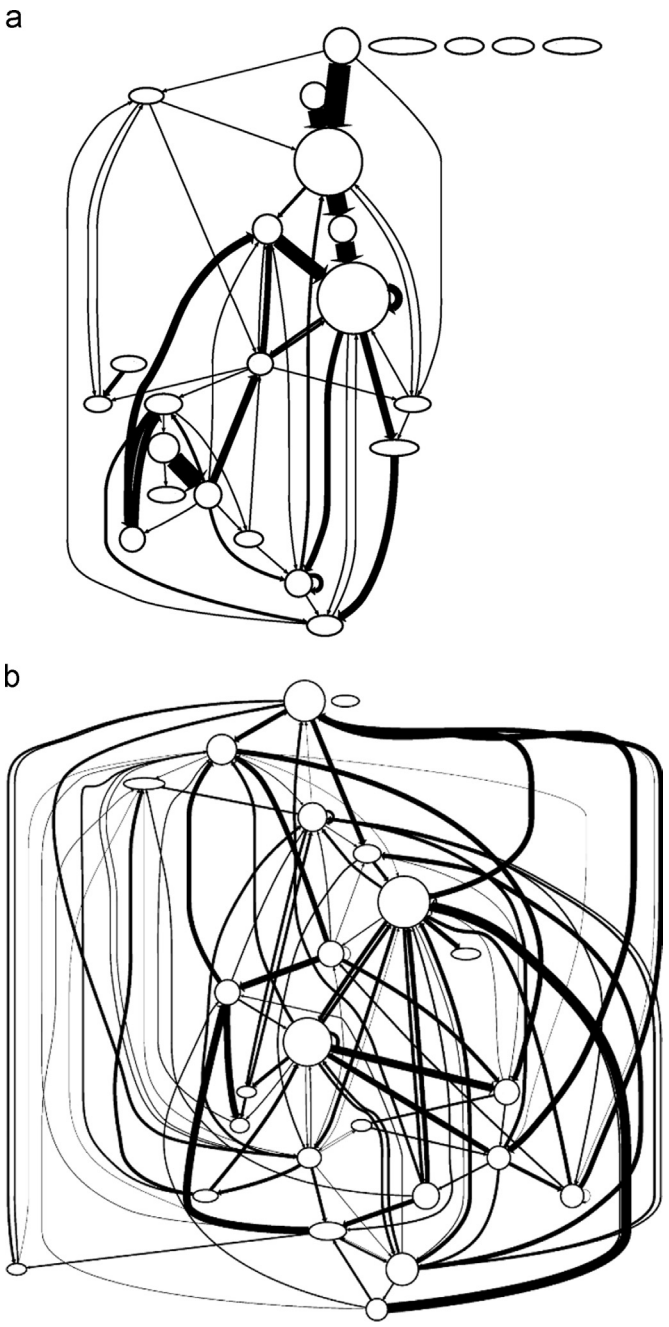


**Fig. 16.** A visualization of the diversity of final trained AdaBoost model without (a) and with (b) speciation.

### 6.4. Important transforms

To test the importance of any given transform, AdaBoost models were made for the various datasets where each transform was excluded in turn. While a very time consuming process it was hoped that certain transforms could be identified as being important for certain datasets. However, no individual transforms were identified as being particularly important. The accuracy could vary to a small degree because a new model was trained and the random nature of how models are built, but no appreciable decrease in accuracy was observed by removing any transform. This indicates that the transforms provide redundant information or, at least, equally significant information. The histogram of oriented gradients transform and the color transform were removed from all the tests performed in this work because they did not improve results and they significantly slowed the computation time of the ECO features algorithm.

### 6.5. Histograms of oriented gradients

The histograms of oriented gradients (HOG) transform is a very popular transform used for object recognition. Due to its popularity in object recognition it was hoped that the addition of the HOG transform would improve the accuracy of ECO features on the datasets used for testing. The other transform in the ECO feature algorithm are considered more basic transforms and the addition of HOG would help determine how more high level transform could perform. The addition did not give better results and slowed the method considerably. In none of the tests reported in this work was the HOG transform used because of this.

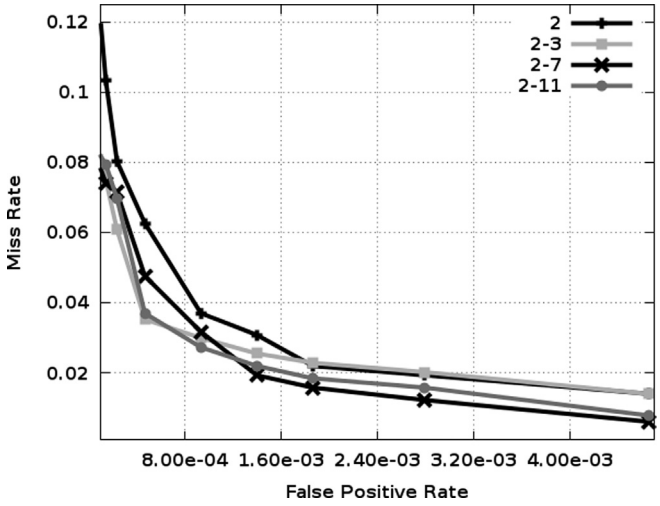


**Fig. 17.** Accuracy of several AdaBoost models where the range of the length of the ECO features used by the models is increased.

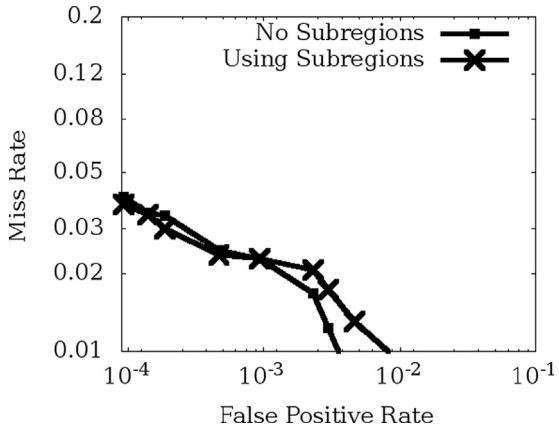


**Fig. 18.** Accuracy of two AdaBoost models where subregions were used and where they were not used on the INRIA person dataset.

## 7. 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 the OpenCV library are used, which are optimized for fast performance.

Most of the training was done on an AMD Phenom II 920 quad-core running at 2.8 GHz, with 4 GB 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 two hours to train altogether. The INRIA Person dataset consisted of a far greater number of images and trained in about two hours 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.

## 8. Conclusion

It was 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. Visualizations of ECO features were given that highlight the information that the ECO features found useful, such as shape information. Speciation allowed the ECO features to perform better by allowing good solutions that took more time to mature to not be discarded by the genetic algorithm and lead to a 5% decrease in miss rate on the INRIA person dataset. The number of transforms of ECO features was further explored and shown that the inclusion of longer transforms does not improve accuracy except for a modest improvement a lower false positive rates. The use of subregions by ECO features was also further explored and shown to not give any accuracy improvements but did lead to a $1.75\times$ speed improvement. AdaBoost was replaced with a support vector machine learning algorithm but did not perform as well as AdaBoost. Finally the importance of transforms was explored and it was found that none of the transforms by themselves were important, indicating that either redundant information or equally significant information is given in the rest of the transforms.

While the ECO features algorithm currently is the best feature construction method for general object recognition there are many improvements that could be made. Beyond what is listed here it is hoped that, based on this work, other intelligent researches will expand the field of feature construction with their own ideas.

The use of a genetic algorithm has a few disadvantages and speciation was added to help deal with this. There are several alternatives that could be used that include evolution strategies, evolutionary programming, simulated annealing, Gaussian adaptation, hill climbing, and particle swarm optimization. It is unclear whether any one of these methods would improve the results, since each method has its advantages, but it would be interesting to look at.

The number of inputs to the perceptrons is fairly high. In many cases the number of inputs to the perceptrons is equal to the number of pixels in the subregion that was selected. With hundreds or thousands of inputs, the perceptron can suffer from Hughes phenomenon, where there are not enough training samples to ensure that there are several examples for each combination of inputs. A higher dimensional feature space will lead to reduced predictive abilities given the same training set. A feature selection of extraction step could be added to reduce the dimensionality of the input space.

There are several things that could be done to improve the runtime performance of ECO features. OpenCV was used extensively throughout this work and many OpenCV functions now also have a GPU implementation. In the same way that we saw a significant improvement in run times by implementing the HOG method on a GPU, a GPU implementation of ECO features could help dramatically. Other works [12,13,64] have avoided using a sliding window approach which could greatly reduce the computations needed when using full images and not just image crops.

## Conflict of interest statement

None declared.

## 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] S. Agarwal, D. Roth, Learning a sparse representation for object detection, in: Proceedings of the 7th European Conference on Computer Vision—Part IV, Springer-Verlag, 2002, pp. 113–130.

[3] H. Motoda, H. Liu, Feature selection, extraction and construction, in: Communication of Institute of Information and Computing Machinery, vol. 5, 2002, pp. 67–72.

[4] M. Andriluka, S. Roth, B. Schiele, Pictorial structures revisited: people detection and articulated pose estimation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2009, pp. 1014–1021.

[5] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (4) (2002) 509–522.

[6] S.M. Bileschi, Streetscenes: Towards Scene Understanding in Still Images, Ph.D. Dissertation, MIT, 2006.

[7] 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.

[8] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, 2010, pp. 1627–1645.

[9] I. Laptev, Improvements of object detection using boosted histograms, in: Proceedings of British Machine Vision Conference (BMVC), 2006.

[10] N.-S. Vu, H.M. Dee, A. Caplier, Face recognition using the poem descriptor, Pattern Recognition 45 (7) (2012) 2478–2488.

[11] H. Schneiderman, T. Kanade, Object detection using the statistics of parts, International Journal of Computer Vision 56 (3) (2004) 151–177.

[12] S. Shah, S. Srinivasan, S. Sanyal, Fast object detection using local feature-based SVMs, in: Proceedings of the IEEE Workshop on Multimedia Data Mining, 2007, pp. 1–5.

[13] L. Spinello, A. Macho, R. Triebel, R. Siegwart, Detecting pedestrians at very small scales, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 4313–4318.

[14] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2001, pp. 511–518.

[15] B. Wu, H. Ai, C. Huang, S. Lao, Fast rotation invariant multi-view face detection based on real AdaBoost, in: Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004, May 2004, pp. 79–84.

[16] X. Yu, Y. Li, C. Fermuller, D. Doermann, Object detection using a shape codebook, in: British Machine Vision Conference (BMVC), vol. 4, 2007.

[17] K. Lillywhite, D. Lee, Automated fish taxonomy using Evolution-COnstructed features, in: Advances in Visual Computing, Lecture Notes in Computer Science, vol. 6938, Springer Berlin/Heidelberg, 2011, pp. 541–550.

[18] K. Lillywhite, B. Tippetts, D. Lee, Self-tuned Evolution-COnstructed features for general object recognition, in: Pattern Recognition, vol. 45, 2012, pp. 241–251.

[19] K. Lillywhite, D. Lee, B. Tippetts, Improving Evolution-COnstructed features using speciation, in: IEEE Workshop on Applications of Computer Vision, Breckenridge, Colorado, USA, January 9–11, 2012 (6 pp.).

[20] H. Liu, H. Motoda, Computational Methods of Feature Selection, Chapman & Hall, 2008.

[21] A.J. Ferreira, M.A. Figueiredo, An unsupervised approach to feature discretization and selection, Pattern Recognition 45 (9) (2012) 3048–3060.

[22] P. Narendra, K. Fukunaga, A branch and bound algorithm for feature subset selection, in: IEEE Transactions on Computers, vol. C-26, September 1977, pp. 917–922.

[23] D. Liu, G. Hua, P. Viola, T. Chen, Integrated feature selection and higher-order spatial feature extraction for object categorization, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2008, pp. 1–8.

[24] X. Wang, J. Yang, X. Teng, W. Xia, R. Jensen, Feature selection based on rough sets and particle swarm optimization, Pattern Recognition Letters 28 (4) (2007) 459–471.

[25] W. Pedrycz, S.S. Ahmad, Evolutionary feature selection via structure retention, in: Expert Systems with Applications, no. 0, 2011.

[26] C. Huang, C. Wang, A GA-based feature selection and parameters optimization for support vector machines, in: Expert Systems with Applications, vol. 31, 2006, pp. 231–240.

[27] J. Bala, K. Jong, J. Huang, H. Vafaie, H. Wechsler, Using learning to facilitate the evolution of features for recognizing visual concepts, in: Evolutionary Computation, vol. 4, MIT Press, 1996, pp. 297–311.

[28] P. Dollár, Z. Tu, P. Perona, S. Belongie, Integral channel features, in: British Machine Vision Conference (BMVC), 2009, pp. 1–11.

[29] Z. Sun, G. Bebis, R. Miller, Object detection using feature subset selection, Pattern Recognition 37 (11) (2004) 2165–2176. Available ⟨http://www.sciencedirect.com/science/article/pii/S0031320304001359⟩.

[30] L. Cao, K. Chua, W. Chong, H. Lee, Q. Gu, A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine, Neurocomputing 55 (1–2) (2003) 321–336.

[31] W. Zuo, D. Zhang, J. Yang, K. Wang, BDPCA plus LDA: a novel fast feature extraction technique for face recognition, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 36 (August (4)) (2006) 946–953.

[32] 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 (October 2007) 1700–1715.

[33] J. Sherrah, R. Bogner, A. Bouzerdoum, The evolutionary pre-processor: automatic feature extraction for supervised classification using genetic programming, in: Proceedings of the 2nd International Conference on Genetic Programming, 1997, pp. 304–312.

[34] M. Smith, L. Bull, Genetic programming with a genetic algorithm for feature construction and selection, in: Genetic Programming and Evolvable Machines, vol. 6, Springer, 2005, pp. 265–281.

[35] K. Neshatian, M. Zhang, M. Johnston, Feature construction and dimension reduction using genetic programming, in: Lecture Notes in Computer Science, vol. 4830, Springer, 2007, pp. 160–171.

[36] H. Vafaie, K. De Jong, Feature space transformation using genetic algorithms, in: IEEE Intelligent Systems and their Applications, 13, March 1998, pp. 57–65.

[37] S. Brumby, J. Theiler, S. Perkins, N. Harvey, J. Szymanski, J. Bloch, M. Mitchell, Investigation of image feature extraction by a genetic algorithm, in: Proceedings of SPIE, vol. 3812, 1999, pp. 24–31.

[38] M. Roberts, E. Claridge, Cooperative coevolution of image feature construction and object detection, in: Lecture Notes in Computer Science, Springer, 2004, pp. 902–911.

[39] V. Bulitko, G. Lee, I. Levner, L. Li, R. Greiner, Adaptive image interpretation: a spectrum of machine learning problems, in: Proceedings of the ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining, 2003, pp. 1–8.

[40] K. Krawiec, B. Bhanu, Visual learning by evolutionary and coevolutionary feature synthesis, IEEE Transactions on Evolutionary Computation 11 (2007) 635–650.

[41] D.C. Ciresan, U. Meier, J. Schmidhuber, Multi-column Deep Neural Networks for Image Classification, CoRR, vol. abs/1202.2745, 2012.

[42] A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems 25, 2012, pp. 1106–1114.

[43] Q. Le, W. Zou, S. Yeung, A. Ng, Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 3361–3368.

[44] A. Mohan, C. Papageorgiou, T. Poggio, Example-based object detection in images by components, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (4) (2001) 349–361.

[45] D. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[46] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, June 2005, pp. 886–893.

[47] S. Maji, A. Berg, J. Malik, Classification using intersection kernel support vector machines is efficient, in: IEEE Conference on Computer Vision and Pattern Recognition, 23–28 June, 2008, pp. 1–8.

[48] T. Serre, L. Wolf, T. Poggio, A new biologically motivated framework for robust object recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[49] C. Siagian, L. Itti, Rapid biologically-inspired scene classification using features shared with visual attention, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (February (2)) (2007) 300–312.

[50] M. Mitchell, An Introduction to Genetic Algorithms, The MIT press, 1998.

[51] R. Fergus, P. Perona, A. Zisserman, Object class recognition by unsupervised scale-invariant learning, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, 2003, pp. 264–271.

[52] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, Robust object recognition with cortex-like mechanisms, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, March 2007, pp. 411–426.

[53] 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.

[54] O. Tuzel, F. Porikli, P. Meer, Pedestrian detection via classification on Riemannian manifolds, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, October 2008, pp. 1713–1727.

[55] M. Norouzi, M. Ranjbar, G. Mori, Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 2735–2742.

[56] M. Burl, L. Asker, P. Smyth, U. Fayyad, P. Perona, L. Crumpler, J. Aubele, Learning to recognize volcanoes on Venus, in: Machine Learning, vol. 30, Springer, 1998, pp. 165–194.

[57] Caltech Computation Vision Group, Caltech Image Datasets, Available ⟨http://www.vision.caltech.edu/html-files/archive.html⟩.

[58] National Institute for Research in Computer Science and Control, INRIA Person Dataset, 2005, Available ⟨http://pascal.inrialpes.fr/data/human/⟩.

[59] P. Dollár, B. Babenko, S. Belongie, P. Perona, Z. Tu, Multiple component learning for object detection, in: European Conference on Computer Vision, October 2008, pp. 211–224.

[60] A. Asuncion, D. Newman, UCI Machine Learning Repository, 2007, Available ⟨http://www.ics.uci.edu/~mlearn/MLRepository.html⟩.

[61] D. Rosebrock, M. Rilk, J. Spehr, F.M. Wahl, Using the shadow as a single feature for real-time monocular vehicle pose determination, in: Advances in Visual Computing, Lecture Notes in Computer Science, vol. 6938, Springer, 2011, pp. 563–572.
[62] S.W. Mahfoud, Niching Methods for Genetic Algorithms, Ph.D. Dissertation, University of Illinois at Urbana-Champaign, 1995.
[63] M. Potter, K. De Jong, Evolving neural networks with collaborative species, in: Summer Computer Simulation Conference, 1995, pp. 340–345.
[64] C. Lampert, M. Blaschko, T. Hofmann, Beyond sliding windows: object localization by efficient subwindow search, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008, pp. 1–8.

**Kirt Lillywhite** received his PhD from the Department of Electrical and Computer Engineering at Brigham Young University, Provo, Utah, in 2012. His research interests include real-time image processing, pattern recognition, parallel processing, and robotic vision. He is currently doing vision research at Smart Vision Works, LLC.

**Dah-Jye Lee** received his B.S. degree from National Taiwan University of Science and Technology in 1984, M.S. and PhD 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 11 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.

**Beau Tippetts** received his PhD from the Department of Electrical and Computer Engineering at Brigham Young University, Provo, Utah, in 2012. 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 doing vision research at Smart Vision Works.

**James Archibald** received the B.S. degree in mathematics from Brigham Young University, Provo, UT, in 1981 and the M.S. and PhD degrees in computer science from the University of Washington, Seattle, in 1983 and 1987, respectively. He has been with the Department of Electrical and Computer Engineering, Brigham Young University since 1987. His research interests include robotics, multiagent systems, and machine vision. Dr. Archibald is a member of the ACM and Phi Kappa Phi.