Pattern Anal Applic (2015) 18:451–459
DOI 10.1007/s10044-014-0426-2

INDUSTRIAL AND COMMERCIAL APPLICATION

Automatic fish taxonomy using evolution-constructed features for invasive species removal

Dong Zhang · Kirt D. Lillywhite · Dah-Jye Lee · Beau J. Tippetts

Received: 1 March 2013 / Accepted: 20 October 2014 / Published online: 2 November 2014

Abstract Assessing the taxonomy of fish is important to manage fish populations, regulate fisheries, and remove the exotic invasive species. Automating this process saves valuable resources of time, money, and manpower. Current methods for automatic fish monitoring rely on a human expert to design features necessary for classifying fish into a taxonomy. This paper describes a method using evolution-constructed (ECO) features to automatically find features that can be used to classify fish species. Rather than relying on human experts to build feature sets to tune their parameters, our method uses simulated evolution to construct series of transforms that convert the input signal of raw pixels of fish images into high-quality features or features that are often overlooked by humans. The effectiveness of ECO features is shown on a dataset of four fish species where using fivefold cross validation an average classification rate of 99.4 % is achieved. Although we use four fish species to prove the feasibility, this method can be easily adapted to new fauna and circumstances.

Keywords Object detection · Feature construction · ECO features · AdaBoost · Fish taxonomy

D. Zhang
School of Information Science and Technology, Sun Yat-Sen University, Guangzhou, China

K. D. Lillywhite · B. J. Tippetts
Smart Vision Works International, Orem, UT, USA

D.-J. Lee (✉)
Department of Electrical and Computer Engineering, Brigham Young University, 459 CB, Provo 84602, UT, USA
e-mail: djlee@byu.edu

1 Introduction

Assessing the taxonomy of fish has become an important issue for managing fish populations, regulating fisheries, and especially for the task of removing invasive species. Reports show that many lakes and rivers of the USA are threatened by invasive fish species. A notable example is the Asian carp which was imported to the southern USA to keep aquaculture facilities clean and to provide fresh fish for fish markets [1]. Since their escape into the wild in the 1980s, Asian carp have been swimming northward and present a threat to the Great Lakes because they directly compete for food with valuable native fish, disturb ecological balances, and even affect environmental and economic systems. Investigation shows that the establishment of Asian carps could cause great economic impact on the Great Lakes' commercial, tribal, and sport fisheries, valued at more than US$7 billion annually [2]. To address exotic species' invasions, an estimated US$120–137 billion is expended annually by the government and commercial entities in the USA [3].

To address the impact of invasive species and reduce their populations, commercial fishing techniques have been implemented in some locations. However, all such operations are labor-intensive and must manually sort out any native species that are also incidentally netted. An automated system for fish monitoring and classification of invasive species could have a huge impact on the way that biologists, government agencies, and commercial fishers operate.

Advances in hardware and image processing as well as pattern recognition methods in the last two decades have made computer vision a widely employed technology in aquaculture [4]. Computer vision-based automatic systems have been proposed for almost all aquaculture operations

and inspections, including counting [5, 6], size measurement and mass estimation [7, 8], gender detection and quality inspection [9, 10], and monitoring welfare [11, 12]. Many of these systems have shown promising results.

Research in the area of assessing taxonomy of fish has also been published. Zion et al. [13] use an underwater sorting machine to sort three species of fish in a fishery pond. They use features that they designed by hand from fish silhouettes and a Bayes classifier to determine the species. Rova et al. [14] use deformable template matching and a support vector machine to differentiate between two very similarly shaped fish. Lee et al. [15] compared contour segments between fish and a database to identify four target species. Chambah et al. [16] use hand-selected shape, color, texture features, motion features, and a Bayes classifier to identify fish in an aquarium. Cadieux et al. [17] use silhouettes and again a set of hand-selected features with a combination of a Bayes classifier, a learning vector quantization, and a neural network to classify fish.

Although previous research methods for automated fish identification and taxonomy have achieved promising results, they have depended on a human expert to design the features the identification algorithm uses. Adapting to other environments with a different fauna is difficult, time consuming, and costly. 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. 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 [18]. 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 [19]. In this paper, we proposed modifying and adapting our previous work on evolution-constructed (ECO) features to automatically find features that are then used by AdaBoost to classify different fish species [20]. Using ECO features allows the system to discover good and useful features to discriminate fish species without the use of a human expert. It is capable of constructing non-intuitive features that are often overlooked by human experts. Although we use four species of fish from the dataset to prove the feasibility, this method can be easily adapted to new fauna and circumstances.

The rest of this paper is structured as follows. Section 2 explains the ECO features in detail. The dataset and experimental results are presented to show the feasibility of this method in Sect. 3. To demonstrate how the ECO works, the visualization of ECO features and discussion are provided in Sect. 4. Finally, our conclusions are given in Sect. 5.

2 ECO features

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, $\mathbf{V}$ is the ECO feature output vector, n the number of transforms the feature is composed of, T_i the transformation at step i, $\mathbf{V}_i$ the intermediate value at step i, φ_i the transformation parameter vector at step i and $I(x_1, y_1, x_2, y_2)$ a subregion of the original image, I, indicated by the pixel range x_1, y_1, x_2, y_2.

$$\begin{aligned} \mathbf{V} &= T_n(\mathbf{V}_{n-1}, \varphi_n) \\ \mathbf{V}_{n-1} &= T_{n-1}(\mathbf{V}_{n-2}, \varphi_{n-1}) \\ &\cdots \\ \mathbf{V}_1 &= T_1(I(x_1, y_1, x_2, y_2), \varphi_1). \end{aligned} \tag{1}$$

Almost any image transformation (or image processing technique) 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, varies

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

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 two to eight 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. Figure 1 shows an example 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. Rather than making any assumptions about what the salient regions of the image are and defining the criteria for their selection, the genetic algorithm is used to search for the subregion parameters x_1, y_1, x_2 and 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 located at a single point or small region of an image, whereas global features cover a large region or the entire image [21]. The use of subregions allows each ECO feature to specialize at identifying different aspects of the target object.

2.2 Constructing ECO features

ECO features are constructed using a standard genetic algorithm (GA) [22]. 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

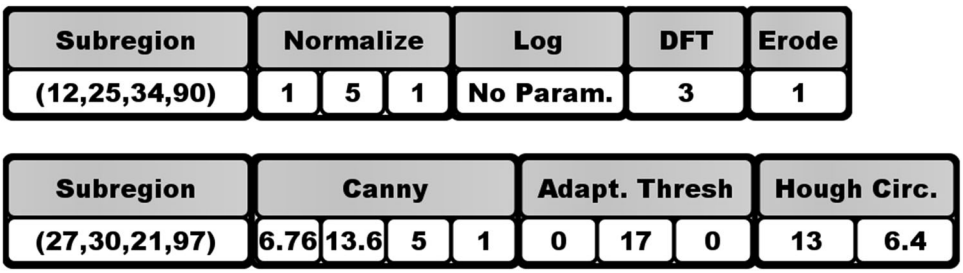

Fig. 1 Two example ECO features. The first example shows an ECO feature where the transforms are applied to the subregion where $x_1 = 12$, $y_1 = 25$, $x_2 = 34$, and $y_2 = 90$ from Eq. 1. The values below the transforms are the parameter vectors φ_i also from Eq. 1

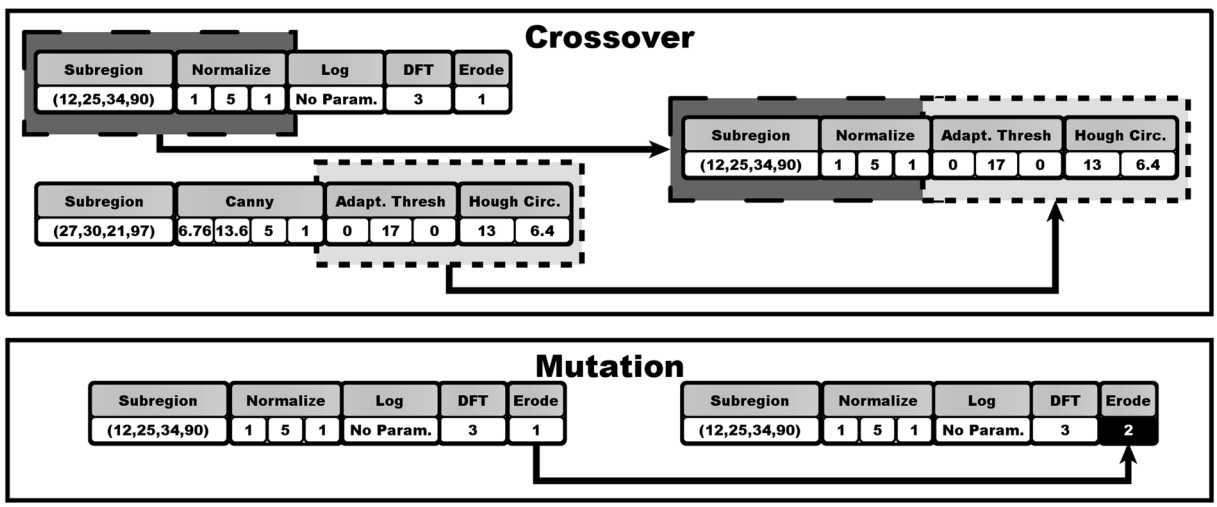

Fig. 2 Examples of crossover and mutation

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 are satisfied.

In our algorithm, GA creatures represent ECO features (examples are shown in Fig. 1). Genes are the elements of an ECO feature which includes 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 make up a creature (or ECO feature) 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. 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 $\mathbf{V}$ to a binary classification, α, through a weight vector $\mathbf{W}$ and a bias term b. If the sample belongs to the specific class that is being detected, α is set to 1. It is set to 0 otherwise:

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

Training the perceptron generates the weight vector $\mathbf{W}$ according to Eq. 3. Training images are processed according to Eq. 1 and the output vector $\mathbf{V}$ 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.

$$\begin{aligned} \delta &= \beta - \alpha \\ \mathbf{W}[i] &= \mathbf{W}[i] + \lambda \times \delta \times \mathbf{V}[i] \end{aligned} \tag{3}$$

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 $[0, 1000]$.

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

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 of creatures with lower fitness scores to move to the next generation can be tuned. Higher values of N mean more creatures compete for one surviving spot and hence 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 (replacing transforms) as shown in Fig. 2. 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 (using different parameters), as also shown in Fig. 2. This whole process of finding features is summarized in Algorithm 1.

Algorithm 1 *Finding ψ Features ψ*

Input: Size of population, images of training set and holding set, set of image transforms, number of generations, bias term b, learning rate λ, parameter for tournament selection N, fitness score threshold;

for Size of population **do**

Randomly create ECO feature. Select $x_1, y_1, x_2, y_2, \mathrm{T}_1(\varphi_1), \ldots, \mathrm{T}_n(\varphi_n)$.

$x_1 \in$ [0, image width-2], $y_1 \in$ [0, image height-2],

$x_2 \in [x_1+1$, image width-1], $y_2 \in [y_1+1$, image height-1],

$\mathrm{T}_1(\varphi_1) \in [\psi], \ldots$, $\mathrm{T}_n(\varphi_n) \in [\psi]$,

$\varphi_1 \in [\xi_{T_1}], \ldots$, $\varphi_n \in [\xi_{T_n}]$.

end for

for number of generations **do**

for every ECO feature **do**

for every training image **do**

Process image with feature transforms

Train creature's perceptron

end for

for every holding set image **do**

Process image with feature transforms

Use perceptron output to update fitness score

end for

Assign fitness score to the creature

Save creature if fitness score $>\psi$threshold

end for

Select creatures to go to next generation

Create new creatures using crossover

Apply mutations to the population

end for

Output: ECO features and corresponding perceptrons

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. 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 equal to 1.0. 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,

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

Algorithm 2 *TrainψAdaBoostψ*

Input: Set of training images M, ECO features and corresponding perceptrons, the maximum number of weak classifiers X;

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, Ω

  if $\delta_w[\Omega] >= 50\%$ **then**

    BREAK

  end if

  Calculate coefficient of perceptron using Equation 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

Output: coefficients corresponding to ECO features ρ.

2.4 Using the AdaBoost model

For every single class, to classify it from other classes, at least one ECO Feature must be constructed. Each ECO feature accompanies its perceptron works as a weak classifier. All ECO features are then combined into one stronger classifier by AdaBoost for classification. Figure 3 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). 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 adjusted to vary the trade-off between false positives and false negatives:

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

3 Experiment and results

The fish images used are from field study images taken by the biology department of Brigham Young University. 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. 4. There are 246 Yellowstone cutthroat, 121 cottids, 140 speckled dace, and 174 whitefish in this dataset. The raw images were pre-processed to make a dataset appropriate for object recognition. The image was rotated so that the head of the fish was on the right side of the image. Then, each image in the fish dataset was cropped and resized to standard 161 × 46 pixels. No color information was used, because in many fish species' recognition applications, color is either not present or not reliable due to water opaqueness and inability to control lighting conditions. The ECO features then have to key into shape information to distinguish one species from another.

Fivefold 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 fivefold cross validation, onefold is treated as the test set and the remaining fourfold is used for training the ECO features. For each fold and species, 10–16 ECO features are found before error rates rise above

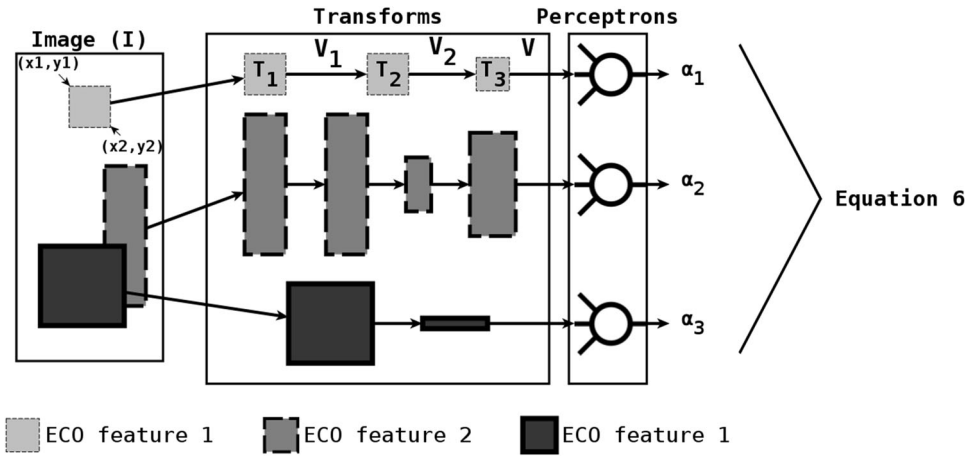

Fig. 3 ECO features and their corresponding perceptrons are combined using AdaBoost to classify an image as object or non-object

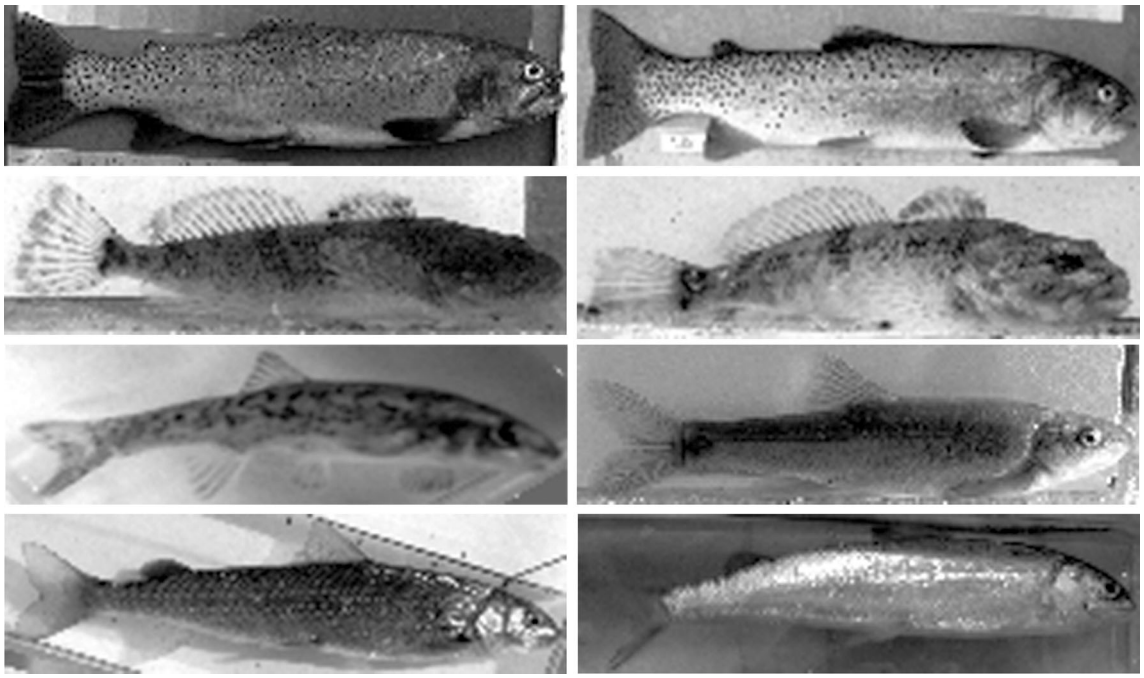

Fig. 4 Examples of the four fish species. From the *top row* to the *bottom row*, the species are Yellowstone cutthroat, cottid, speckled dace, and whitefish

50 % during AdaBoost training. Once the ECO features were found, the images in the current fold were used to compute the classification accuracy. Table 2 shows the classification accuracy when doing fivefold cross validation (onefold for testing and four for training) for each species. The first column, for example, lists the classification accuracy values when using the samples in Fold-1 for test and the samples in Fold-2 to -5 for training, while the value in the first row of first column shows the classification accuracy when Yellowstone Cutthroat is chosen as positive sample and the other three species as negative samples.

The results are given in Table 2 with the average classification accuracy being 99.4 %, and a standard deviation of 0.64 %. This shows that the ECO features performed very well in discriminating between the four fish taxonomies. It took approximately 3 min using a desktop computer to find the ECO features for a single fold during cross validation. Finding the ECO features is an off-line process and only needs to be run as the location and fauna change, as new species are added, or some other phenomena occur that change the environment. This makes using ECO features very adaptable and easy to use, without sacrificing the accuracy of the system. Another advantage of ECO features is that it does not require human experts to determine what are significant features for classification. Compared with our previous works, which depended on a human

Table 2 The classification accuracy when doing fivefold cross validation (onefold for testing and four for training) for each species

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

One species is treated as the positive example and the other species form the negative examples

expert to design the features [23], the method proposed in this paper is able to construct efficient and non-intuitive features that are often overlooked by human experts.

4 Discussion

As has been presented, the performance of ECO features on fish classification is promising, and generalizes well across BYU fish datasets. However, the entire process works like a "black box". As stated previously, the main advantage of ECO features is that they can be constructed automatically without any manual process or human intervention. To prove that our theory is valid and the whole ECO features concept is efficient, we turned our focus 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 transform, $\mathbf{V}_i$ (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. 7. 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 finds:

$$\mathbf{V}_i(x,y) = \frac{\mathbf{V}_i(x,y) - \min(\mathbf{V}_i)}{\max(\mathbf{V}_i) - \min(\mathbf{V}_i)} \tag{7}$$

The output of the final transform, $\mathbf{V}$, becomes the input to the perceptron of the ECO feature. Those inputs are multiplied by the perceptron weights, $\mathbf{W}$. The greater the magnitude of the perceptron weight, the more important is the input that is connected to that weight. 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.

Figure 5 shows a visualization of four ECO features that have been trained on the BYU fish dataset using the species whitefish as the positive examples and the other three species as the negative examples. The visualizations show the shape information that is extracted and separates the fish species from each other.

ECO feature A performs an adaptive threshold and then resizes the image to make it smaller. To make it easier to be viewed in Fig. 5, the visualization of the resize transform and the perceptron weights are shown in the original size before the resize transform. ECO feature A looks at the shape of the head and some of the shape of the body as evidenced in the visualization of the positive perceptron weights. ECO feature B performs a Gabor filter followed by a single erode transform along a long strip across the spine of the fish. The result of the Gabor filter and erode transform is a silhouette of that part of the fish. From the silhouette, the positive perceptron weights are strongest when the dorsal fin is in a particular position and the negative weights are strongest when the adipose fin is in a particular position.

ECO feature C takes a fairly large subregion of the fish that includes basically all of the fish except the head. It first performs a difference of Gaussians that really emphasizes the shape of the fish along the spine. It then performs a census transform and then a Harris corner strength transform. The positive perceptron weights show a bright spot at the point where the tail connects to the rest of the body, indicating that its position is important. The negative weights show if an anal fin shows up at a certain location in the image that it is not a whitefish. ECO feature D performs a Harris corner strength transform with a large window size followed by a discrete Fourier transform. Because of the Fourier transform, the visualization is hard to interpret. The dimensionality of the output image and of the perceptron is high, but very little of it contains useful information. ECO features do not attempt to reduce the dimensionality of its output.

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

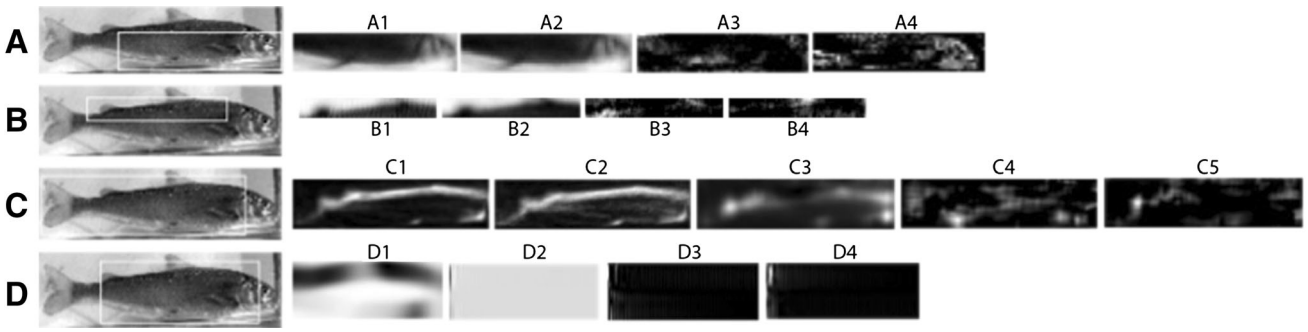

Fig. 5 Visualization of four ECO features trained on the BYU fish dataset. Transforms are as follows *A.1* adaptive threshold, *A.2* resize, *B.1* Gabor filter, *B.2* erode, *C.1* difference of Gaussians, *C.2* census, *C.3* Harris corner strength, *D.1* Harris corner strength, and *D.2* discrete Fourier transform. *A.3*, *B.3*, *C.4*, and *D.3* are visualizations of the negative magnitude perceptron weights and *A.4*, *B.4*, *C.5*, and *D.4* are the positive magnitude perceptron weights

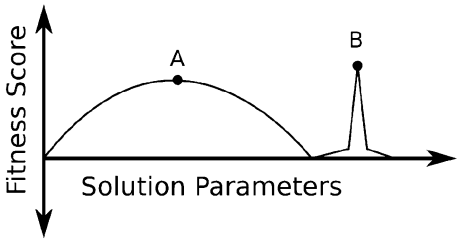

Fig. 6 A search algorithm to find the maximum fitness score is much more likely to find *point A* before *point B*

solutions, but if given the opportunity could eventually evolve to have a similar or even better fitness score. For example, Fig. 6 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 the hill of fitness score provided around solution A. Searches that start off the hill around point A could find point B, which has a slightly higher fitness score, but takes 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 [24–26].

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 do not divide the ECO features arbitrarily. A simpler method to provide speciation, which 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.

5 Conclusion

This paper employs ECO features to provide an effective way to perform fish taxonomy classification. Compared with our previous works, which depended on a human expert to design the features the identification algorithm uses, the method proposed in this paper is able to construct efficient and non-intuitive features that are often overlooked by human experts. No human expert was needed to design the features used to discriminate between fish taxonomies. If new fish species are added to the environment, no human expert is needed to design a new feature set. If a radically different shaped fish is introduced, new ECO features can be found automatically rather than waiting for an expert to make the necessary changes. Our experiments showed that ECO features obtained an average of 99.4 % classification accuracy with a standard deviation of 0.64 % on our proposed dataset of four distinct fish species. The time needed to set up the system to recognize the fish taxonomies is minimal. Despite the advantage of constructing useful and meaningful features automatically, the recognition accuracy of our method is not compromised.

References

1. Asian Carp Control, The official website of the Asian Carp Regional Coordinating Committee. http://asiancarp.org/background.asp

2. ASA (2008) Today's angler: a statistical profile of anglers, their targeted species and expenditures. American Sportfishing Association, Alexandria
3. USDA, RMRS Invasive Species Research. http://www.rmrs.nau.edu/invasive_species/index.php
4. Zion B (2012) The use of computer vision technologies in aquaculture—a review. Comput Electron Agric 88:125–132
5. Alver MO, Tennoy T, Alfredsen JA, Oie G (2007) Automatic measurement of rotifer *Brachionus plicatilis* densities in first feeding tanks. Aquacult Eng 36(2):115–121
6. Zheng X, Zhang Y (2010) A fish population counting method using fuzzy artificial neural network. In: Proceedings of International Conference on Progress in Informatics and Computing (PIC), IEEE, pp 225–228
7. Beddow TA, Ross LG, Marchant JA (1996) Predicting salmon biomass remotely using a digital stereo-imaging technique. Aquaculture 146:189–203
8. Zion B, Ostrovsky V, Karplus I, Lidor G, Barki A (2012) Ornamental Fish Mass Estimation by Image Processing. Agricultural Research Organization, Bet Dagan
9. Karplus I, Alchanatis V, Zion B (2005) Guidance of groups of guppies (Poecilia reticulata) to allow sorting by computer vision. Aquacult Eng 32:509–520
10. Wallat GK, Luzuriaga DA, Balaban MO, Chapman FA (2002) Analysis of skin color development in live goldfish using a color machine vision system. N Am J Aquacult 64:79–84
11. Duarte S, Reig L, Oca J (2009) Measurement of sole activity by digital image analysis. Aquacult Eng 41:22–27
12. Rodriguez A, Bermudez M, Rabunal JR, Puertas J, Dorado J, Pena L, Balairon L (2011) Optical fish trajectory measurement in fishways through computer vision and artificial neural networks. J Comput Civ Eng 25:291–301
13. Zion B, Alchanatis V, Ostrovsky V, Barki A, Karplus I (2007) Real-time underwater sorting of edible fish species. Comput Electron Agric 56:34–45
14. Rova A, Mori G, Dill L (2007) One fish, two fish, butterfish, trumpeter: recognizing fish in underwater video. In: Proceedings of IAPR Conference on Machine Vision Applications, IAPR, pp 404–407
15. Lee DJ, Archibald J, Schoenberger R, Dennis A, Shiozawa D (2008) Contour matching for fish species recognition and migration monitoring. Appl Comput Intell Biol 122:183–207
16. Chambah M, Semani D, Renouf A, Courtellemont P, Rizzi A (2004) Underwater color constancy: enhancement of automatic live fish recognition. In: Color Imaging IX: Processing, Hardcopy, and Applications, vol. 5293. pp 157–168
17. Cadieux S, Michaud F, Lalonde F (2000) Intelligent system for automated fish sorting and counting. In: International Conference on Intelligent Robots and Systems, vol 2. pp 1279–1284
18. Li Z, Yang Y, Liu J, Zhou X, Lu H (2012) Unsupervised Feature Selection Using Nonnegative Spectral Clustering. In: Proceedings of the 26th AAAI Conference on Artificial Intelligence, AAAI, pp 1026–1032
19. Lillywhite K, Tippetts B, Lee DJ (2012) Self-tuned evolution-constructed features for general object recognition. Pattern Recogn 45(1):241–251
20. Lillywhite K, Lee DJ (2011) Automated fish taxonomy using evolution-constructed features. In: Advances in Visual Computing, ser. Lecture Notes in Computer Science. vol 6938. Springer, Berlin, pp 541–550
21. Roth P, Winter M (2008) Survey of appearance-based methods for object recognition. Technical report ICGTR0108 (ICG-TR-01/08), Institute for Computer Graphics and Vision, Graz University of Technology
22. Mitchell M (1998) An introduction to genetic algorithms. The MIT press, Cambridge
23. Lee DJ, Archibald JK, Schoenberger RB, Dennis AW, Shiozawa DK (2008) Contour matching for fish species recognition and migration monitoring. Applications of computational intelligence in biology: current trends and open problems. Springer, Berlin. ISBN 978-3-540-78533-0
24. Stanley K, Miikkulainen R (2002) Efficient reinforcement learning through evolving neural network topologies. In: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann Publishers Inc., pp 1757–1762
25. Mahfoud SW (1995) Niching methods for genetic algorithms. Ph.D. dissertation, University of Illinois at Urbana-Champaign
26. Potter M, De Jong K (1995) Evolving neural networks with collaborative species. In: Summer Computer Simulation Conference, Society for computer simulation, pp 340–345