Evolution of Computer Vison Subsystems in Robot Navigation and Image Classification Tasks

Sascha Lange and Martin Riedmiller

Neuroinformatics Group,
Institute for Computer Science and Institute for Cognitive Science,
University of Osnabrück, 49069 Osnabrück, Germany
{Sascha.Lange,Martin.Riedmiller}@uos.de

Abstract. Real-time decision making based on visual sensory information is a demanding task for mobile robots. Learning on high-dimensional, highly redundant image data imposes a real problem for most learning algorithms, especially those being based on neural networks. In this paper we investigate the utilization of evolutionary techniques in combination with supervised learning of feedforward nets to automatically construct and improve suitable, task-dependent preprocessing layers helping to reduce the complexity of the original learning problem. Given a number of basic, parameterized low-level computer vision algorithms, the proposed evolutionary algorithm automatically selects and appropriately sets up the parameters of exactly those operators best suited for the imposed supervised learning problem.

1 Introduction

A central problem within the field of robotics is to appropriately respond to sensory stimulation in real-time. At this, visual sensory information becomes more and more important. When applying techniques of machine learning to such tasks, one faces a number of recurring problems. In most cases it is simply not feasible to feed the whole image information directly into a neural network or into any other learning algorithm. Often a (hand-coded) computer vision subsystem is used to preprocess the images and extract useful visual features. Usually, the developer himself specifies such a restricted set of features and the structure of an intermediate data representation. By using the resulting low-dimensional feature vectors and the intermediate data representation, learning becomes possible.

Since there is still no all-purpose domain-independent computer vision system, adapting or redesigning the computer vision subsystem for a new problem is a time consuming and expensive task. But compared to finding a general-purpose computer vision subsystem and a transferable intermediate data representation it is generally easier and more reliable to build highly specialized, task-dependent preprocessing layers. Accordingly, we believe computer vision subsystems should concentrate on extracting only task-relevant features.

What features are needed to be extracted in order to be fed to the learning algorithm, is highly dependent on the task itself. Therefore, when specifying the intermediate data representation and designing the computer vision subsystem, it is absolutely necessary to consider not only the input images but also the desired output signals.

To give an example (see experiment 2): Consider a set of some images containing a dice in front of a uniform background and others containing only the empty background. If the task was to detect the presence or absence of the dice in a particular image it would at least be necessary to look for a rectangular or quadratic area and return a boolean value. If the dice should be grabbed by a robot-arm the detected feature ("quadratic area") could still be the same. Additionally it would however be necessary to calculate its center of gravity or its bounding box. Finally, if the task was to read the dice it would be necessary to detect a completely different feature, namely the markers on it. However, in this task the position of the features is unimportant and does not need to be considered. Encoding their individual absence or presence is sufficient. It does not take much effort to come up with scenarios in which an extraction of even more features is needed.

This example clearly shows that different applications require different sets of features to be extracted. Searching the images for prominent or interesting features without considering the application of the extracted information can not give the optimal solution to a task. The approach presented in this paper, therefore is to treat the learning of both the preprocessing and the control subsystem as a whole, always considering both input and desired output.

The algorithm presented here, is able to directly solve a supervised learning problem on visual input without any further information or help from the outside. We have chosen to use a hybrid solution that comprises an evolutionary algorithm (outer loop) and the supervised training of feedforward nets (inner loop). The candidate solutions (individuals of the evolutionary process) are composed of a neural network realizing the decision making process and a specialized and highly task-dependent computer vision subsystem for preprocessing the image data. Candidate preprocessing subsystems are automatically constructed during the evolutionary process. The primitives forming the computer vision subsystem are more or less complex, highly parameterized, hand-coded programs, each realizing a feature detector, global operator or other low-level computer vision algorithm.

2 Related Work

To tackle the problem of feature selection a considerable amount of methods has been developed. Statistical methods based on principle component analysis (PCA) (for example [5], [8], [9]) are able to remove redundant features with a minimal loss of information. One probelm of such methods is that they are often computationally expensive. Locally linear embedding (LLE) is a non-linear noniterative unsupervised method that can be used for dimensionality reduction [15]. LLE can be extended to data with class information available [13], too.

Martin C. Martin has used genetic programming to construct visual feature extracting programs for a computer vision subsystem of a robot solving a simple navigation task [11]. Instead of considering the task as a whole, the subtask to be solved by the visual feature extractors and their correct output was explicitly specified.

Tony Belpaeme proposes a genetic algorithm for the evolutionary construction of visual feature extractors not needing any information about the target output of these extractors [2]. According to Belpaeme it would be possible to measure the fitness of the extractors as their performance in the task itself. But for performance reasons he proposes to use the entropy of the resulting feature vectors instead. Belpaeme does not provide any information about the generalization behavior of the algorithm or its performance in real tasks.

The Schema Learning System (SLS) proposed by Bruce A. Draper [6] is able to learn special-purpose recognition strategies under supervision using a library of executable procedures. The parameters of the computer vision algorithms in the library, however, have to be set by hand for every different task [7].

Bala et al. [1] use a genetic algorithm to search for a subset of user provided visual features in an object classification task. A classifier is constructed by inducing a decision tree using these feature subsets and a training set of labeled images.

In [3] Heinrich Braun and Joachim Weisbrod present a hybrid algorithm (ENZO) evolving the topology of feedforward nets. As ENZO is permitted to change the input layer, it is able to select features and reduce the dimensionality. The main fitness criterion is the performance of the evolved nets during a "local" training phase. The algorithm was later combined with reinforcement learning to solve complex learning problems [4].

3 Description of the Algorithm

3.1 Evolution of Candidate Solutions

The navigation and classification tasks examined have all been formulated as supervised learning problems: Given a training set

$$P = \{(\mathcal{B}_p; \boldsymbol{y}_p) \in [0,255]^{3^{k \times l}} \times [0,1]^m \mid p = 1, ..., n\}$$

of n training patterns $(\mathcal{B}_p; \boldsymbol{y}_p)$ the task is to find a control program minimizing an error term. We have chosen to use the "total sum of squares" (tss)

$$E = \sum_{p=1}^{n} \sum_{j=1}^{m} (o_{pj} - y_{pj})^2$$

of the difference between the output $\boldsymbol{o_p}$ of the control program and the target $\boldsymbol{y_p}$. In all of these tasks, the input is a single multi-spectral image encoded in a $k \times l$-matrix $\mathcal{B}$. The entries of the matrix are color values given in the RGB color space.

In order to minimize the error term an evolutionary algorithm is proposed. The algorithm constructs and modifies complex candidates of a predefined architecture.

The Structure of the Individuals. The evolved candidates (individuals $\boldsymbol{x_i}$ of the population X_t) consist of two distinct subsystems. The layer processing the raw image matrices is a low-level computer vision subsystem. The control subsystem forming the second layer receives its input data from this computer vision subsystem. It performs the output generation (action selection) based on a more compact representation of the original data.

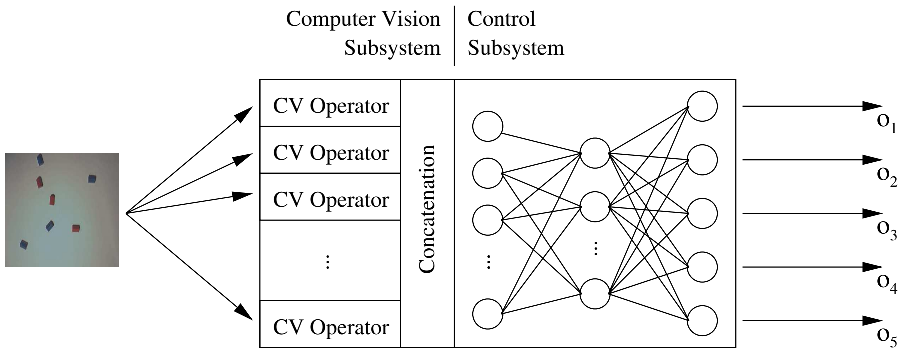

Fig. 1. The general structure of a candidate. The candidate is composed of the computer vision sublayer and a downstream neural network realizing the control subsystem. The computer vision subsystem consists of smaller building blocks (operators) that all access the image matrix directly. Their individual output is concatenated and returned as the output of the computer vision subsystem

The subsystem realizing low-level vision tasks is a set of vertically ordered smaller subsystems that will be called "operators" throughout this text. These operators all realize a parameterized (by parameter vector $\boldsymbol{p}$) function $h_{\boldsymbol{p}}(\mathcal{B}) = \boldsymbol{v}$ returning a q-dimensional vector $\boldsymbol{v}$ for every given image matrix $\mathcal{B}$. These operators could be simple filters like Gaussian blur, point-of-interest detectors like corner detectors or even complex programs like segmentation algorithms. The operators all reduce the input to a representation of the image much simpler and more compact than the image itself. This could be realized by (preferably) reducing the dimension or by choosing a "simpler" but still iconic encoding of prominent features (like edge or region maps) or by combining both of these methods.

Since these operators are ordered in parallel (Fig. 1) they all access the image directly and do not consider the output of each other.

Generally the operators employed in this work need at least one parameter to appropriately adapt their behavior to the circumstances. Some operators may even change their behavior substantially according to different parameter settings. Each operator returns a vector of constant length that is concatenated to the output vectors of the other operators to form the final output of the image preprocessing layer. The length of the output vector is allowed to change only when the parameters are changed. Besides the parameters influencing the behavior of the operators, each operator has some boolean parameters to switch the output of a specific feature on or off.

Concluding the image processing layers of the candidates may differ in their composition of operators, in the setting of parameters controlling the internal behavior of the operators and in the selection of output values returned.

The control layer is formed by a simple feedforward net. The dimension of the input layer is determined by the size of the image processing layer's output vector, whereas the dimension of the output layer is determined by the dimension of the target vector provided with the training patterns. The internal topology of the net, however, is a free variable. Because the topology of a neural net has a major impact on its learning behavior and has to be chosen carefully it will be subject to evolutionary optimization too.

The Evolutionary Algorithm. Starting with a random initialized population, an (μ, λ)-evolutionary strategy with the addition of elitism is followed. The implemented algorithm comprises five different steps: Selection, recombination, mutation, evaluation and reinsertion (see Fig. 2).

During the selection phase, μ parents are chosen randomly for mating. The probability $P(\boldsymbol{x}_i)$ of selecting individual $\boldsymbol{x}_i$ as the k-th parent is proportional to its fitness:

$$P(\boldsymbol{x}_i) = \frac{f(\boldsymbol{x}_i)}{\sum_{\boldsymbol{x}_j \in X_t} f(\boldsymbol{x}_j)}.$$

During the recombination phase every two "neighboring" individuals $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are recombined to form an offsprings $\boldsymbol{x}'$ by randomly choosing a subset of operators from the original operators of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The network topology is passed unchanged from the first individual.

Afterwards a mutation operator is applied to the resulting offsprings. A random number of entries of the computer vision operators' parameter vectors and the topology of the neural network are mutated by adding a zero-mean gaussian random variable. The chance of changing the entry p_i of the parameter vector $\boldsymbol{p}$ is set to 0.1 in all experiments.

Every newly formed individual is evaluated according to the fitness function described in the next section.

Finally, the fittest λ childrens replace the λ worst individuals of population X_t to receive the new population X_{t+1}. This is a combination of elitism and fitness based reinsertion.

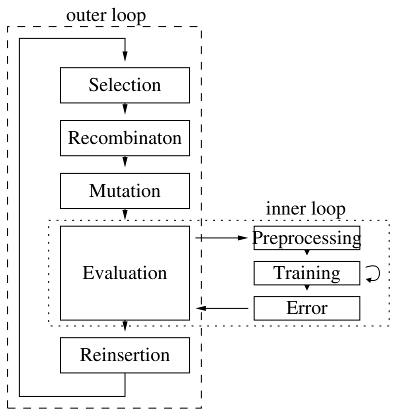

Fig. 2. The different steps of the algorithm. The outer cycle consists of the five steps of the evolutionary algorithm. The inner evaluation cycle evaluates each candidate by using its computer vision subsystems to obtain the preprocessed input data and by training the neural network several epochs on this data. The observed training and testing error are used to calculate the fitness

It should be noted that although the evolution of good solutions typically takes several hours (when not parallelized), the basic operators and the resulting programs are able to analyze the input and calculate an output in real-time on present personal computers.

Training the Net to Measure the Fitness. The images $\mathcal{B}$ of the training patterns $(\mathcal{B}_i; \boldsymbol{y}_i)$ are analyzed with each candidate's vision subsystem to form a set of training patterns $(h(\boldsymbol{\mathcal{B}}_i); \boldsymbol{y}_i)$ suitable for the input layer of its neural network. The net is trained on the resulting pattern set using resilient propagation (Rprop [14]) for a specific number of epochs. Afterwards the neural network is evaluated on the testing set. The tss on the training and testing set are both normalized to yield a fitness value between 0 and 1 (bigger means better). The final fitness of the candidate is the weighted sum of these two values and some additional, less important and lower weighted components that may reward good runtime performance or smaller memory consumption.

In all experiments discussed in this paper, the fitness has been a weighted sum of training error, testing error (smaller gives higher fitness) and of three lower weighted factors penalizing higher numbers of hidden layer neurons and connections, bigger input layers and the runtime needed to process one image by the candidate's vision subsystem. These factors have shown to produce efficient candidates with rather small networks and a good generalization behavior.

3.2 Building Blocks of the Computer Vision Subsystem.

From the huge number of well studied operators, detectors and algorithms, we have selected and adapted or implemented five algorithms to be used in the experiments. These operators are: a corner-detector employing the SUSAN principle [16], an operator returning a single image from the gauss pyramid, a histogram operator, returning n 2-dimensional UV histograms of n non-overlapping image partitions (inspired by [17]), a region-growing algorithm known as "color structure code" (CSC, [12]) and the operator with the most parameters, a segmentation algorithm based on the lower layers of the CVTK library [10].

Whereas two operators return iconic representations (of reduced dimensionality), the other operators return "meaningful" symbolic representations. For example the CVTK and CSC operators both return abstract descriptions of uniformly colored image regions containing information such as color, coordinates of the center of gravity, area, bounding rectangle and compactness of the regions found.

Each operator needs at least one parameter to be set properly; for example: the level of the gauss pyramid to return or the maximum color-distance being used by the CSC algorithm to decide, whether or not two neighboring pixels belong to the same region.

Since the detailed description of all algorithms and parameters would go beyond the scope of this paper, we will exemplarily describe in detail the CVTK operator.

This operator uses a given set S of samples $\boldsymbol{s} = (\boldsymbol{s}_{\text{rgb}}, \boldsymbol{s}_c) = (r, g, b, c)$ of the discrete "classification" function $f : I^3 \mapsto 0, 1, .., N$ that assigns a class label c to every rgb-value $\boldsymbol{s}_{\text{rgb}} \in I^3$. Implementation dependent, the values of the red, green and blue channel are from the set $I = \{0, 1, ...255\}$.

The algorithm uses a simple nearest-neighbor classification to segment the image. The label of each pixel b_{ij} of the image $\mathcal{B}$ is determined by finding the "closest" sample $\boldsymbol{s}_{\text{closest}} = \arg\min_{\boldsymbol{s}_{\text{rgb}}} d(\boldsymbol{s}_{\text{rgb}}, b_{ij})$ and assigning its class label $\boldsymbol{s}_c$ iff the distance is smaller than a threshold t. If there is no sample having a distance smaller than t the class label of the background (0) is assigned to b_{ij}.

The distance d is the Euclidian distance between the color values of the pixel b_{ij} and the sample $\boldsymbol{s}$ after transforming both to one of four possible color spaces (RGB, YUV, UV, CIE L*a*b*). The color space can be chosen by setting a parameter of the operator.

After classifying the whole image pixel by pixel, neighboring pixels of the same color class are connected to regions. Internally, the algorithm uses a contour based region description to calculate region features like center of gravity, area, bounding rectangle or compactness. Finally the algorithm returns a list of n region descriptions. By setting boolean output parameters, the user can specify what features each description should contain. Table 1 lists all parameters of the CVTK operator.

Table 1. Parameters of the CVTK operator. Some parameters are influencing the segmentation process itself, whereas a second group of parameters selects what properties should be returned

	parameter	type	multitude	description
algorithm	max_classes	int	1	number of different color classes
	color_space	int	1	encodes the color space to use
	(r,g,b,c)	int^4	1..n	labeled color samples
	t	double	1	distance threshold
	min_size	double	1	minimum size of regions
output	regions_out	int	1	regions per color class to return
	order	boolean	1	order according to area
	area	boolean	1	return the area
	center	boolean	1	return the center of gravity
	class	boolean	1	return the class label
	compactness	boolean	1	return the compactness
	bounding_box	boolean	1	return corners of bounding box

4 Results

The algorithm has been tested on two classification tasks and on one robot-navigation experiment. We have not used any artificial images but only real world images captured by a digital camera under typical office-lighting conditions. There have been absolutely no efforts to reduce shadows or to ensure uniform lighting conditions. The image sets can be found at http://www-lehre.inf.uos.de/~slange/master/ .

Experiment 1: Subtraction of Colored Gaming Pieces. The images show different settings of up to eight red and blue gaming pieces on a white background. There are always at least as many blue as red pieces and never more than four pieces of the same color in the image frame. The task is to calculate the difference between the number of blue and red pieces and to label the images accordingly. There is an additional testing set that is used after finishing the evolution to judge the generalization performance of the whole evolutionary process – in contrast to the net's testing error that is calculated on the first testing set and used to guide the selection of the individuals. The training set contains 98 images, the testing set 27 images and the second testing set 9 images (320×240 pixel).

Experiment 2: Reading a Dice. In this setup a dice was filmed from a constant camera position. The images have to be classified into 6 classes according to the number of markers on the upper side of the dice. Again, there are three sets of equally sized (160×120 pixel) and correctly labeled images. The training set consists of 45 images, the testing set of 12 and the additional testing set contains 15 images. The second testing set also contains 5 images of a completely different and slightly bigger dice than the dice used during the evolution.

Experiment 3: Driving to the Ball. In this simple navigation task, a camera-equipped robot has to drive to a ball that is positioned somewhere in the surrounding area. There are no obstacles between the robot and the ball. The training and testing data is acquired by hand-steering the robot to the ball with a cableless joystick. The task is solved several times by a human. During this process the program stores a few hundred captured images together with the human provided control signal to form the training and testing sets. Afterwards a controller is evolved on this training data and finally tested in the real environment.

All experiments have been conducted with an initial population size of 60, $\mu = 20$ and $\lambda = 15$. The evolutionary process was always stopped after 200 epochs.

As a "proof of concept" the subtraction experiment has been solved first by allowing the candidate computer vision subsystems to contain only a single CVTK operator. This operator has a really huge parameter space and has to be set up carefully since the provided sample colors completely determine the result of the segmentation process. Due to changing lighting conditions and shadows the task of finding good sample colors is very difficult even for a human.

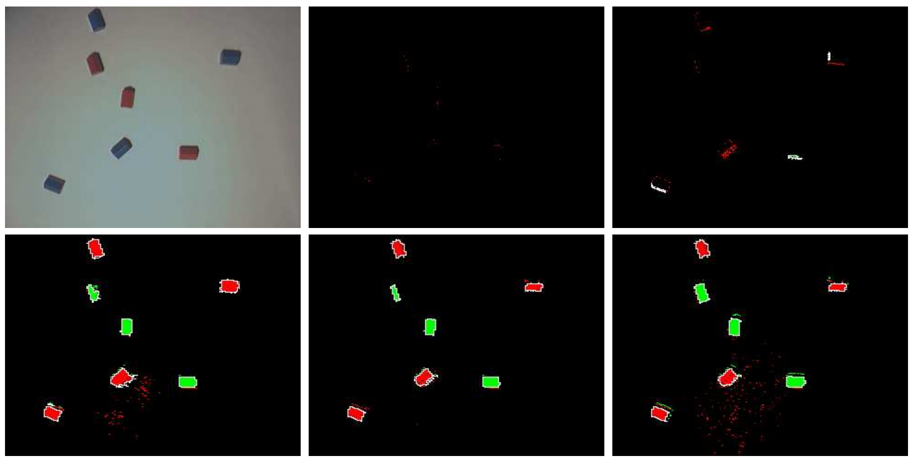

Fig. 3. From top to bottom, from left to right: Orinal image of the testing set of the subtraction task and the region image produced by the single cvtk operator of the best individual after $t = 0$, $t = 12$, $t = 25$, $t = 50$, und $t = 150$ epochs

The progress made during the evolution in extracting the interesting regions is visualized in fig. 3. While the image processing is not substantially improved after the 50th epoch, the other settings keep improving.

Actually, in the very first run of the experiment, the best individual after 200 epochs adapted a quite intelligent and efficient strategy. Inspecting the parameter settings and generated intermediate representations, we have found the operator to not only robustly find the gaming pieces but also to return the smallest feature vector possible: The individual discards any information about the position, size and area and only considers the color of the regions found. The feature vector passed from the computer vision subsystem to the learning subsystem has only eight "boolean" entries – four for each of the two foreground color classes. This small representation is possible because the necessary minimal size of regions was set appropriately during evolution to filter out all false positive detections, that are consistently smaller in size than the correctly detected regions.

In all of the five repetitions we have performed with this setup, the best individual after 200 epochs always classified at least eight of the second, unseen testing set's nine images correctly. As can be seen in fig. 4 the training error of the best individual is very low right from the beginning. This is no suprise because a sufficiently big feedforward net is easily trained to memorize a relatively small training set. Obviously, it is the generalization error that has to be minimized during the evolutionary process.

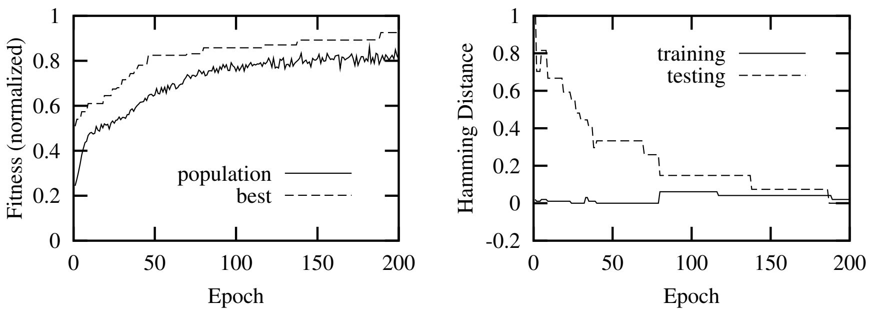

Fig. 4. Results of training a single CVTK operator in experiment 1. The fitness of the population and its best individual improve parallely (left). The Hamming Distance measured for the best individual continously decreases on the testing set (right). The Hamming Distance has beend divided by the number of patterns in order to be comparable

Afterwards, the algorithm was tested in the first two experiments with all operators active. The classification results listed in tab. 2 show the resulting solutions to perform clearly above chance.

During the experiments, the algorithm was observed to have problems in the first task due to some operators dominating the entire population too early. This happens, because the population size – due to the computational cost – has been chosen to be relatively small and some operators may give notably better results than others during early stages of the optimization. If those "early starting" operators are not the operators that give the better results in the long run, the problem might occur.

This problem might be circumvented by simply training the different operators separatly in the early stages of the evolutionary process. Table 2 shows that evolving the operators for the first 30 epochs separately solves the observed problems effectively.

Table 2. Percentage of the correct classifications of the images of the two testing sets by the best individual after 200 epochs of evolution. The evolution has been started for 0, 10, 30 epochs with isolated subpopulations each containing only one type of operators

	Subtraction		Dice	
	testing set	testing set 2	testing set	testing set 2
no isolation	96%	44%	100%	100%
10 epochs isolated	96%	56%	100%	93%
30 epochs isolated	100%	100%	100%	100%

Finally, the algorithm was tested successfully in the robot-navigation task[1]. Although the ball is nearly always reached, the robot drives very slowly at some positions. We believe this behavior results from inaccurate or contradictory control signals in the training pattern rather from an error in the evolved computer vision subsystem.

One interesting observation from the inspection of the vision systems constructed is that instead of detecting the whole area of the ball, some subsystems only search for an equatorial stripe. The center of the detected stripe always closely coincides with the center of the ball. The advantage of extracting this smaller region seems to be that problems due to highlights in the upper region of the ball and shadows in the lower half could be effectively circumvented. Compared to the subsystems which considered the whole area of the ball, the subsystems seem to be more robust against noisy pixels in the background and therefore had to sort out fewer false detections. In spite of this obvious difference in the preprocessing layer we were not able to detect any significant differences in the behavior of the robot.

[1] Multimedia files of both the training and testing phase can be found at http://www-lehre.inf.uos.de/˜slange/master/.

5 Conclusion

We have introduced an elegant algorithm that is able to directly learn different tasks on visual input. In contrast to earlier work, it does not need any a priori knowledge neither about features to be extracted nor the layout of an intermediate representation. It is able to construct specialized, task-dependent computer vision subsystems that enable a learning algorithm to successfully learn the task. Moreover it finds good parameter settings for the employed operators even in huge parameter spaces. The generalization performance of the resulting strategies is clearly above chance. As yet, we have only used tasks having a "color-based" solutions. We plan to implement other operators and to try different and more difficult tasks in the future.

References

1. Bala, J. , DeJong K., Huang, J., Vafaie, H., Wechsler, H.: Hybrid Learning Using Genetic Algorithms and Decision Tress for Pattern Classification. 14th Int Joint Conf. on Artifical Intelligence (IJCAI), Canada (1995) 719-724
2. Belpaeme, T.: Evolution of Visual Feature Detectors. In: Proceedings of the First European Workshop on Evolutionary Computation in Image analysis and Signal Processing (EvoIASP99, Göteborg, Sweden), University of Birmingham School of Computer Science technical report. (1999)
3. Braun, H., Weisbrod, J.: Evolving feedforward neural networks. Proc. of the International Conference on Artificial Neural Nets and Genetic Algorithms (1993)
4. Braun, H., Ragg, T.: Evolutionary Optimization of Neural Networks for Reinforcement Learning Algorithms. In: ICML96, Workshop Proceedings on Evolutionary Computing and Machine Learning, Italy (1996) 38-45
5. McCabe, G.P.: Principal Variables. In: Technometrics vol. 26 (1984) 127-134
6. Draper., B.: Learning Object Recognition Strategies. Ph.D. dissertation, Univ. of Massachusetts, Dept. of Computer Science. Tech. report 93-50 (1993).
7. Draper, B.: Learning Control Strategies for Object Recognition. In: Visual Learning, Ikeuchi and Veloso (eds.). Oxford University Press (1996)
8. Jolliffe, I.T.: Principal Component Analysis. Springer-Verlag, Berlin Heidelberg New-York (1986)
9. Krzanowski, W.J.: Selection of Variables to Preserve Multivariate Data Structure, Using Principal Component Analysis. In: Applied Statistics- Journal of the Royal Statistical Society Series C vol. 36 (1987) 22-33
10. Lange, S.: Verfolgung von farblich markierten Objekten in 2 Dimensionen. B.Sc. thesis, Universität Osnabrück, Institut für Kognitionswissenschaft (2001)
11. Martin, C. M.: The Simulated Evolution of Robot Perception. Ph.D. dissertation, Carnegie Mellon University Pittsburgh (2001)
12. Priese, L., Rehrmann, V., Schian R., Lakmann, R.: Traffic Sign Recognition Based on Color Image Evaluation. In: Proc. Intelligent Vehicles Symposium (1993) 95-100
13. De Ridder, D., Kouropteva, O., Okun, O., Pietikäinen, M. and Duin, R.P.W.: Supervised locally linear embedding. In: Proc. Joint Int. Conf. ICANN/ICONIP 2003, Lecture Notes in Computer Science, vol. 2714, Springer Verlag, Berlin Heidelberg New York (2003) 333-341

14. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: the Rprop algorithm. In: Proceedings of the ICNN, San Francisco (1993)
15. Roweis, S. T., Saul, L. K.: Nonlinear dimensionality reduction by locally linear embedding. In: Science, 290(5500) (2000) 2323-2326
16. Smith, S. M.: A new class of corner finder. In: Proc. 3rd British Machine Vision Conference (1992) 139-148
17. Steels, L., Kaplan, F.: AIBO's first words. The social learning of language and meaning. In: Gouzoules, H. (ed) Evolution of Communication, vol. 4, nr. 1, Amsterdam: John Benjamins Publishing Company (2001)