ACTIVE OBJECT DETECTION

G. de Croon
MICC-IKAT, Universiteit Maastricht, P.O. Box 616, 6200 MD, Maastricht, The Netherlands

Keywords: Object Detection, Active Vision, Active Scanning, Evolutionary Algorithms.

Abstract: We investigate an object-detection method that employs active image scanning. The method extracts a local sample at the current scanning position and maps it to a shifting vector indicating the next scanning position. The method's goal is to move the scanning position to an object location, skipping regions in the image that are unlikely to contain an object. We apply the active object-detection method (AOD-method) to a face-detection task and compare it with window-sliding object-detection methods, which employ passive scanning. We conclude that the AOD-method performs at par with these methods, while being computationally less expensive. In a conservative estimate the AOD-method extracts 45 times fewer local samples, leading to a 50% reduction of computational effort. This reduction is obtained at the expense of application generality.

1 INTRODUCTION

Object detection is the automatic determination of image locations at which instances of a predefined object class are present. Numerous methods for object detection exist (e.g., (Viola and Jones, 2001; Fergus et al., 2006)), most of which scan a part of the image at some stage of the object-detection process. Until now, this scanning is performed in a passive manner: local image samples extracted during scanning are not used to guide the scanning process. We mention two main object-detection approaches that employ passive scanning here. The window-sliding approach to object detection (e.g., (Viola and Jones, 2001)) employs passive scanning to check for object presence at all locations of an evenly spaced grid. This approach extracts a local sample at each grid point and classifies it either as an object or as a part of the background. The part-based approach to object detection (e.g., (Fergus et al., 2006)) employs passive scanning to determine interest points in an image. This approach calculates an interest-value for local samples (such as entropy of gray-values at multiple scales (Kadir and Brady, 2001)) at all points of an evenly spaced grid. At the interest points, the approach extracts new local samples that are evaluated as belonging to the object or the background. Although some methods try to limit the region of the image in which passive scanning is applied (e.g., (Murphy et al., 2005)), it remains a computationally expensive and inefficient scanning method: at each sampling point computationally costly feature extraction is performed, while the probability of detecting an object or suitable interest point can be low.

In this article, we investigate an object detection method that employs active scanning (based on (de Croon and Postma, 2006)). In active scanning local samples are used to guide the scanning process: at the current scanning position a local image sample is extracted and mapped to a shifting vector indicating the next scanning position. The method takes successive samples towards the expected object location, while skipping regions unlikely to contain the object. The goal of active scanning is to save computational effort, while retaining a good detection performance. In a companion article, we address the importance of our approach for Embodied Cognitive Science (de Croon and Postma, 2007). In this article we focus on the practical applicability in computer vision. In particular, we verify whether the method reaches its goal for a face-detection task studied before in (Kruppa et al., 2003; Cristinacce and Cootes, 2003). We compare the method's performance and computational complexity with that of the object detectors (belonging to the window-sliding approach) employed in the previous studies.

The rest of the paper is organised as follows. In Section 2, we introduce the object-detection method.

Then, in Section 3, we explain our experimental setup. In Section 4 we analyse the results of the experiments. We draw our conclusions in Section 5.

2 ACTIVE OBJECT-DETECTION METHOD

The active object-detection method (AOD-method) scans the image for multiple discrete time steps in order to find an object. In our implementation of the AOD-method this process consists of three phases: (i) scanning for likely object locations on a coarse scale, (ii) refining the scanning position on a fine scale, and (iii) verifying object presence at the last scanning position with a standard object detector. Both the first and the second phase are executed by an 'agent' that extracts features from local samples, and maps these features to scanning shifts in the image. We refer to the agent of the first phase as the 'remote' agent and to the agent of the second phase as the 'near' agent.

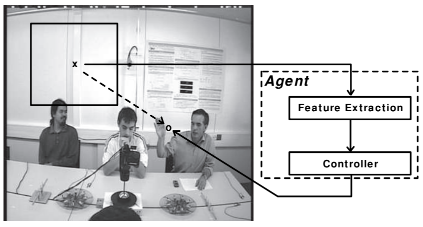

Figure 1: One time step in the scanning process.

Figure 1 illustrates one time step in the scanning process. At the first time step ($t = 0$) of a 'run', the remote agent takes a local sample at an initial, random location in the image ('x' in the figure). The local sample consists of the gray-values in the scanning window. First, the agent extracts features from this local sample. Then, its controller transforms these features to a scanning shift in the image (dashed arrow) that leads to a new scanning location ('o'). We do not allow the scanning window to leave the image. On the next time step ($t = 1$), at the new scanning location, the process of feature extraction and shifting is repeated. The sequence of sampling and shifting continues until $t = T$, where T is an experimental parameter. The goal of the remote agent is to center the local sampling window on an object at $t = T$. Because the remote agent does not always succeed in its goal, we employ a near agent. It starts scanning at the final scanning position of the remote agent and makes scanning shifts until $t = 2T$. At $t = 2T$ we verify object presence at the final scanning position with a standard object detector, such as the one in (Viola and Jones, 2001).

3 EXPERIMENTAL SETUP

3.1 Agent Implementation

We first discuss the feature extraction and then the controller. We adopt the integral features introduced in (Viola and Jones, 2001). These features represent contrasts in mean light intensity between different areas in an image. The main advantage of these features is that they can be extracted with very little computational effort, independent of their scale. Figure 2 shows the types of features that we use in our experiments. We illustrate an example of a feature in the bottom of Figure 2. The feature is of type 1 and spans a large part of the right half of the scanning window. The value of this feature is equal to the mean gray-value of all pixels in area A minus the mean gray-value of all pixels in area B. The example feature will respond to vertical contrasts in the image. Since all gray-values are in the interval $[0,1]$, the feature value is in the interval $[-1,1]$.

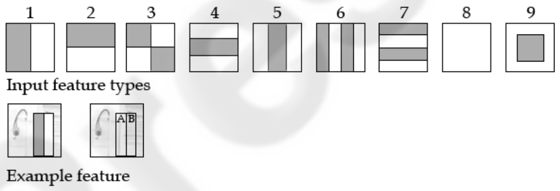

Figure 2: Feature types (top) and example feature (bottom).

We extract n features from the sampling window. They form a vector that serves as input to the controller, which is a completely connected multilayer feedforward neural network. The network has h hidden neurons and $o = 2$ output neurons, all with a sigmoid activation function: $f(x) = \tanh(x)$. The two output neurons encode for the scanning shift $(\Delta x, \Delta y)$ in pixels as follows: $\Delta x = \lfloor o_1 j \rfloor$, $\Delta y = \lfloor o_2 j \rfloor$. The constant j represents the maximal displacement in the image in pixels.

3.2 Evolutionary Algorithm

We employ a 'μ, λ' evolutionary algorithm (Bäck, 1996) to select the features and optimise the neural network weights of both the remote and the near agent, for the following two reasons. First, an evolutionary algorithm can optimise both the controller and the feature extraction simultaneously. Second, an evolutionary algorithm optimises the controller over the entire chain of samples and actions, from $t = 1$ to $t = T$, enabling the agent to employ non-greedy scanning policies. We first evolve the remote agent for uniformly distributed starting positions, and then the near agent in the following manner. We measure the average distance of the evolved remote agent to the nearest object at $t = T$ in the images of the training set. Then, we evolve the near agent for positions that are normally distributed with as mean an object position and as standard deviation the measured average distance at $t = T$.

We split evolution in two by evolving both the features and the neural network weights in the first half, and evolving only the neural network weights in the second half. Evolution starts with a population of λ different agents. An agent is represented by a vector of real values (doubles), referred to as the genome. In this genome, each feature is represented by five values, one for the type and four for the two coordinates inside the scanning window. Each neural network weight is represented by one value. We evaluate the performance of each agent on the task by letting it perform R runs per training image, each of T time steps. The fitness function we use in the first half of evolution is: $f_1(a) = (1 - \text{distance}(a)) + \text{recall}(a)$, where $\text{distance}(a) \in [0,1]$ is the normalised distance between the agent's scanning position at $t = T$ and its nearest object, averaged over all training images and runs. The term $\text{recall}(a)$ is the average proportion of objects that is detected per image by an ensemble of R runs of the agent a. An object is detected if the scanning position is on the object. When all agents have been evaluated, we test the best agent on the validation set. In addition, we select the μ agents with highest fitness values to form a new generation. Each selected agent has λ/μ offspring. To produce offspring, there is a p_{co} probability that one-point cross-over occurs with another selected agent. Furthermore, the genes of the new agent are mutated with probability p_{mut}. The process of fitness evaluation and procreation continues for G generations. As mentioned, we stop evolving the features at $G/2$. In addition, we set p_{co} to 0, since cross-over might be disruptive for the optimisation of neural network weights (Yao, 1999). Moreover, we gradually diminish p_{mut}. Finally, we also change the fitness function f_1 to $f_2(a) = \text{recall}(a)$. At the end of evolution, we select the agent that has the highest weighted sum of its fitness on the training set and validation set (according to the set sizes) to prevent overfitting.

The near agent is evolved in exactly the same manner as the remote agent, except for the different starting positions (close to the objects) and the fitness function: $g(a) = (1 - \text{distance}(a)) + \text{precision}(a)$, which does not change at $G/2$. $\text{precision}(a)$ is the proportion of runs R of the near agent that detect objects at the end of the run. The goal of the near agent is to refine the scanning position reached by the remote agent, by detecting the nearest object and approaching its center as much as possible.

The third phase of the AOD-method, the object detector that verifies object-presence at the last scanning position, is not evolved, but trained according to the training scheme in (Viola and Jones, 2001).

3.3 Face-detection Task

We apply the AOD-method to a face-detection task that is publicly available. We use the FGNET video sequence (http://www-prima.inrialpes.fr/FGnet/), which contains video sequences of a meeting room, recorded from two different cameras. For our experiments we used the joint set of images from both cameras ('Cam1' and 'Cam2') in the first scene ('ScenA'). The set consists of 794 images of 720×576 pixels, which we convert to gray-scale. We use the labelling that is available online, in which only the faces with two visible eyes are labelled. For evolution, we divide the image set in two parts: half of the images is used for testing and half of the images for evolution. The images for evolution are divided in a training set (80%), and a validation set (20%). We perform a two-folded test to obtain our results, and run one evolution per fold.

3.4 Experimental Settings

Here we provide the settings for our experiments. The maximal scanning shift j is equal to half the image width for the remote agent, and equal to one third of the image width for the near agent. The scanning window is a square with sides equal to one third of the image width for the remote agent, and one fourth of the image width for the near agent. The number of time steps per agent is $T = 5$, and the number of runs per image R is 20. We use $n = 10$ features that are extracted from the sampling window. We set the number of hidden neurons h of the controller to $n/2 = 5$, while the number of output neurons o is 2. We set the evolutionary parameters as follows: $\lambda = 100$, $\mu = 25$, $G = 300$, $p_{\text{mut}} = 0.04$, and $p_{\text{co}} = 0.5$.

4 RESULTS

4.1 Behaviour of the Evolved Agents

In this subsection, we give insight into the scanning behaviour of the remote and near agents evolved on the first fold (their behaviour on the second fold is similar). Figure 3 shows ten independent runs of the remote agent. At $t = 0$, all runs are initialised at random positions in the image. The method then successively takes samples and makes scanning shifts (arrows). At the end of scanning ($t = T$) seven out of ten runs have reached an object location. The final locations of the runs are indicated with circles.

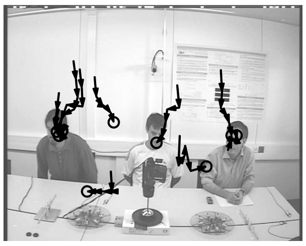
Figure 3: Ten independent runs of the remote agent.

The figure indicates that the evolutionary algorithm found suitable features and neural network weights for the remote agent. Figure 4 shows the ten evolved features inside the scanning window (white box) centered in the image ('x' indicating the scanning location) and the types, sizes and locations of the features. Although it is not straightforward to interpret the features, we can see that it contains both coarse contextual features (e.g., features 2 and 9) and more detailed object-related features (e.g., features 3, 5, and 8).

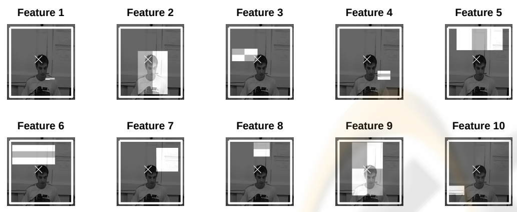

Figure 4: The ten evolved features of the remote agent.

The controller maps these features to scanning shifts that approach the target objects. In the left part of Figure 5, we illustrate the function of the remote agent's controller by taking local samples at a fixed grid, and visualising both the direction and size of the scanning shifts. The controller defines a gradient map on the image that has attractors at persons and at heads in particular. Few of the arrows go upwards. This property of the behaviour is mainly due to two factors. First, the prior distribution of object locations is such that faces usually occur in the lower half of images. Second, the fitness function of the remote agent promotes recall. Since the agent is evaluated on the ensemble of R runs, it can 'loose' a few runs in the bottom of the image as long as the other runs are successful. This raises the issue whether the method exploits more than the prior distribution of face-locations. The AOD-method cannot exploit this prior distribution directly, since it only uses visual features. However, it can exploit it indirectly for samples that contain little information on the object position. Indeed, in the face images, the remote agent seems to have a preference for moving down instead of up. However, the method can move up if the features contain enough information (see the arrows under the face of the standing person). In addition, in (de Croon and Postma, 2006), a different version of the AOD-method performed well on a task of license-plate detection in which there was no strong prior distribution of object locations. In the right part of Figure 5, we show the scanning behaviour of the near agent, close to an object. The near agent considerably improves performance on the detection task, as will be shown in the next subsection.

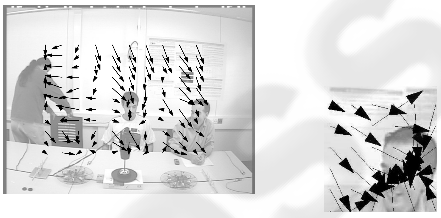
Figure 5: Remote agent's actions at different locations (left) and near agent's actions close to a face (right).

4.2 Performance Comparison

For a possible computational efficiency of the AOD-method to be relevant, the active object-detection method must at least have a sufficient performance. Figure 6 shows an FROC-plot of our experimental results (square markers, thick lines), for the remote agent alone (solid line), the remote and near agent in sequence (dashed line), and the sequential agent followed by the first stage of a Viola and Jones-detector trained according to the training scheme in (Viola and Jones, 2001) (dotted line, only based on the first testing fold). We created these FROC-curves by varying the number of runs: $R = 1, 3, 5, 10, 20$, and 30. In addition, the figure shows the results on the FGNET image set from other studies, made by varying the classifier's threshold: from (Cristinacce and Cootes, 2003) (thin lines), of a Fröba-Küllbeck detector (Fröba and Küllbeck, 2002) ('+'-markers) and a Viola and Jones detector (Viola and Jones, 2001) ('o'-markers). It also shows the results of two Viola and Jones detectors trained on a separate image set and tested on the FGNET set (Kruppa et al., 2003) (thick lines). The first of these detectors attempts to detect face regions in the image, as the detectors in (Cristinacce and Cootes, 2003) ('o'-markers). The second of these detectors attempts to detect a face by including a region around the face, including head and shoulders ('x'-markers).

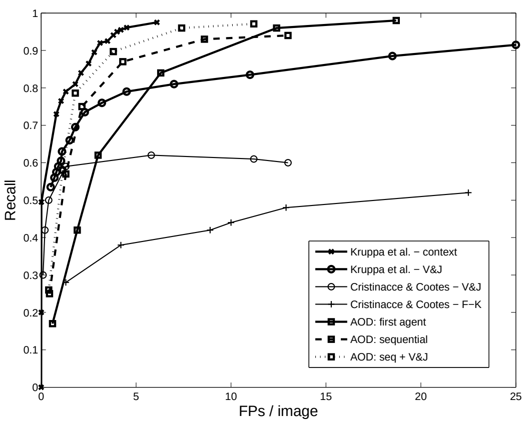

Figure 6: FROC-plot of the object-detection methods.

Figure 6 shows that the AOD-method outperforms the window-sliding approaches that did not include a face's context for detection rates higher than 65%. Detecting faces without considering context is difficult in the FGNET video-sequence, because the appearance of a face can change considerably from image to image (Cristinacce and Cootes, 2003). However, the context of a face (such as head and shoulders) is rather fixed. This is why approaches that exploit this context (such as (Kruppa et al., 2003)) have a more robust performance. The active object-detection method exploits context even to a greater extent than in (Kruppa et al., 2003) that only includes a small area around the object.

The difference between the Viola and Jones-classifier used in (Cristinacce and Cootes, 2003) and (Kruppa et al., 2003) can be explained by at least three factors: the different training set, different parameter settings of the training method for the Viola and Jones-classifier, and a different labeling. In (Kruppa et al., 2003) profile faces are also labeled, while such faces are not labeled in the labeling available online. Small differences between the experiments aside, the results show that the AOD-method performs at par with other existing object detection methods on the FGNET face-detection task.

4.3 Computational Efficiency

4.3.1 General Comparison

The computational costs C of a window-sliding approach (WS) and an active object-detection approach (AOD) can be expressed as:

$$C_{\text{WS}} = G_{\text{H}} G_{\text{V}} (F_{\text{WS}} + Cl) + P \quad (1)$$

$$C_{\text{AOD}} = R(2T)(F_{\text{AOD}} + Ct) + R(F_{\text{WS}} + Cl) + P \quad (2)$$

The variables G_{H} and G_{V} are the number of horizontal and vertical grid points, respectively. Furthermore, F_{WS} is the number of operations necessary for feature extraction in the window-sliding approach, Cl for the classifier, and P for preprocessing. For the AOD-approach, R is the number of independent runs and $2T$ the number of time steps at which local samples are used for scanning shifts. F_{AOD} is the number of operations necessary for feature extraction, and Ct for the controller that maps the features to scanning shifts. $R(F_{\text{WS}} + Cl)$ is for verifying object-presence at the final scanning position.

The AOD-approach is computationally more efficient than the window-sliding approach. The main reason for this is that the AOD-approach extracts far fewer local samples, i.e., $(R(2T) + R) \ll G_{\text{H}} G_{\text{V}}$, while its feature extraction and controller do not cost much more than the feature extraction and classifier of the window-sliding approach. For example, in the FGNET task a window-sliding approach that verifies object presence at every point of a grid with a step size of two pixels will extract $335 \times 248 = 83,080$ local samples (based on the image size, the average face size of 50×80 pixels, and the largest step size mentioned in (Viola and Jones, 2001)). In contrast, the AOD-method extracts $R(2T + 1) = 20 \times 11 = 220$ local samples (sequential agent in combination with a classifier). Under these conditions, the window-sliding approach extracts 377.6 times more local samples than the AOD-method[1].

4.3.2 Estimate of Computational Effort

We estimate the computational effort of both methods for the face-detection task, expressed in a number of operations. We make a conservative estimate for the window-sliding method, the Viola and Jones detector (Viola and Jones, 2001). Importantly, we make the conservative assumption that $G_{\text{H}} = G_{\text{V}} = 100$, which implies step sizes of $\Delta x \approx 7$ and $\Delta y \approx 5$ in the FGNET images. As a result, the AOD-method extracts 45 times fewer local samples. In our estimate of the Viola and Jones detector, we base ourselves on the research in (Cristinacce and Cootes, 2003; Kruppa et al., 2003; Viola and Jones, 2001). We estimate the remaining variables in equation 1 and 2 as follows:

- $F_{\text{WS}} = 64, F_{\text{AOD}} = 80$: In (Viola and Jones, 2001), the computational cost of feature extractions was expressed in array references. The features in Figure 2 require from 4 to 12 references, with as average ≈ 8 references. The average *number* of features extracted per scanning

[1]Note that taking into account different scales of object detection would imply a new multiplication factor for the computational costs, which is disadvantageous for the window-sliding approach.

location was mentioned to be 8 in (Viola and Jones, 2001), while it is 10 for the AOD-method.

- $Cl = 9, Ct = 94$: The classifier of the window-sliding approach makes a linear combination of the features and compares this to a threshold, and therefore we estimate its cost at the average number of features extracted plus one: $8+1=9$. In the neural network of the AOD-method, each hidden and output neuron computes a linear combination of its inputs and puts the result into the activation function: $Ct = h(n+1) + o(h+n+1) + (h+o) = 94$. The first two terms represent the computational costs for the linear combinations made in the hidden and output neurons, respectively. The last term, $(h+o)$, represents the cost for the activation functions.
- $P = 414,720$: Both methods need to calculate an 'integral image' (see (Viola and Jones, 2001)) for their subsequent feature extractions. The computational cost of the calculation of the integral image is a pass through all pixels of the image, being $414,720$ pixels for the 720×576 images.

These estimates lead to $C_{\text{WS}} = 1,144,720$ and $C_{\text{AOD}} = 450,980$: the application of active scanning roughly results in a reduction of 50% of the computational effort. Note that the calculation of the integral image constitutes the main part of the computational costs for the AOD-method. The low number of samples of the AOD-method opens up possibilities for the real-time application of features that are in themselves more costly per sample than integral features, but that require no detailed preprocessing of the entire image.

4.4 Application Generality

The advantages of the AOD-method come at the expense of application generality. Namely, they rely on the exploitation of the steady properties of an object's context. If these properties are present in the test images, the method is still able to detect objects. For example, Figure 7 shows how the remote agent behaves if it is applied to photos taken in our own office (10 independent runs of the remote agent per image). Each scanning shift is represented by an arrow, while the last scanning position has a circle. The run of the remote agent is followed by the first stage of a Viola and Jones classifier, where the runs shown in black belong to runs for which the last scanning position was classified as an object position. In both images the independent runs cluster at the heads, since the office walls are relatively uncluttered (with an occasional poster) as in the FGNET video-sequence. However, if the exploited contextual properties are not present (as in many outdoor images) detection performance degrades considerably. The question is how limiting this loss of generality is. Findings on the human visual system (Henderson, 2003) suggest that this limitation may be relieved by extending the AOD-method, so that it applies different scanning policies to different visual scenes.

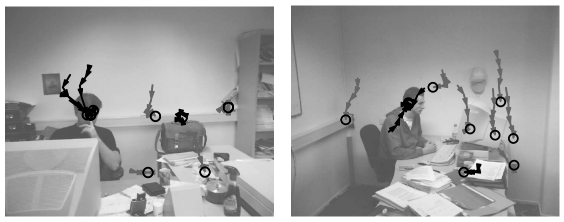
Figure 7: Generalisation of the evolved method.

5 CONCLUSION

We conclude that the AOD-method meets its goal on the FGNET face-detection task: it performs at par with existing object-detection methods, while being computationally more efficient. In a conservative estimate the active object-detection method extracts 45 times fewer local samples than a window-sliding method, leading to a 50% reduction of the computational effort. The advantages of the AOD-method derive from the exploitation of an object's context and come at the cost of application generality.

REFERENCES

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York, Oxford.

Cristinacce, D. and Cootes, T. (2003). A comparison of two real-time face detection methods. In 4th *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance,* pages 1–8.

de Croon, G. and Postma, E. O. (2006). Active object detection. In *Belgian-Dutch AI Conference, BNAIC 2006, Namur, Belgium*.

de Croon, G. and Postma, E. O. (2007). Sensory-motor coordination in object detection. In *First IEEE Symposium on Artificial Life.*

Fergus, R., Perona, P., and Zisserman, A. (in press - 2006). Weakly supervised scale-invariant learning of models for visual recognition. *International Journal of Computer Vision*.

Fröba, B. and Küllbeck, C. (2002). Robust face detection at video frame rate based on edge orientation features. In 5th *international conference on automatic face and gesture recognition 2002,* pages 342–347.

Henderson, J. M. (2003). Human gaze control during real-world scene perception. *TRENDS in Cognitive Sciences*, 7(11).

Kadir, T. and Brady, M. (2001). Scale, saliency and image description. *International Journal of Computer Vision*, 45(2):83–105.

Kruppa, H., Castrillon-Santana, M., and Schiele, B. (2003). Fast and robust face finding via local context. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS'03), Nice, France*.

Murphy, K., Torralba, A., Eaton, D., and Freeman, W. (2005). Object detection and localization using local and global features. In *Sicily workshop on object recognition*. Lecture Notes in Computer Science.

Viola, P. and Jones, M. J. (2001). Robust real-time object detection. *Cambridge Research Laboratory, Technical Report Series*.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87:1423 – 1447.