Sensory-motor Coordination in Object Detection

G. de Croon and E.O. Postma
MICC-IKAT, Universiteit Maastricht
Email: g.decroon@micc.unimaas.nl, postma@micc.unimaas.nl

Abstract

In this article we investigate whether an artificial agent can successfully perform the complex task of object detection in static natural images. For this task, we evolve situated agents that map local image samples to gaze shifts in order to find object locations in the image. We apply the agents to both a task of mug detection in a domestic environment and a task of face detection in an office environment. The analysis of evolved agents shows that they employ sensory-motor coordination to exploit the object's visual context. The experimental results show that the situated agents achieve a detection performance equal to that of existing object detection methods, while extracting ∼50 times fewer local image samples. This advantage comes at the expense of limited generalisation performance: evolved agents exploit the scene-specific contextual clues which may be confined to a single type of visual environment and may therefore not generalise to other types of visual environments. We conclude that the studied situated agents can efficiently and successfully perform object detection at the cost of application generality.

1 Introduction

Embodied Cognitive Science (ECS) represents a bottom-up approach to artificial intelligence and has focused for a great deal on minimally cognitive agents. Central to ECS is the concept of sensory-motor coordination: an agent's coordination of its closed loop of actions and sensory inputs. For example, sensory-motor coordination has been shown to simplify classification tasks [1, 2, 3]. Although ECS research gradually progresses towards applying agents to more complex tasks [4], sceptics have doubts on whether this bottom-up approach to artificial intelligence will scale up. For example, Edelman criticises Beer's research in [5] on categorisation through sensory-motor coordination, by stating that *"the target of the analysis - the evolved solution to a toy task - seems to be hardly worth the effort."* [6]. The goal of this article is to address this criticism by developing and applying an ECS-model to the complex task of object detection in static natural images. By developing a closed-loop model that employs sensory-motor coordination we aim to show that ECS can contribute to more complex cognitive tasks.

The existing, open-loop computer vision approaches to object detection (e.g., [7, 8, 9, 10, 11]) are still outperformed by human vision. A partial explanation for this might lie in the fact that sensory-motor coordination plays a central role in human vision [12], whereas it does not in an open-loop approach. This suggests that it might be advantageous to employ sensory-motor coordination in a computer vision approach to object detection. In this article, we focus on the following research question: *Can an artificial agent successfully perform the task of object detection in static natural images by means of sensory-motor coordination?* We consider an agent successful, if it can perform at least as good as existing open-loop object detection methods on the same task.

To answer the research question, we evolve and study situated agents (inspired by [13, 14, 15]) whose sensory inputs are local image samples and whose actions are gaze shifts in the image. The goal of the agents is to center their gaze on object locations. To test whether the situated agents can be applied to different object detection tasks, we apply them to two tasks: the detection of mugs in a domestic environment and the detection of faces in an office environment. The main characteristic of both tasks is that the visual context of the objects is relatively constant. We selected the tasks on this characteristic, since the agent's sensory-motor apparatus enables it to exploit the visual context for localising an object. We use the mug-detection task to investigate whether and how evolved situated agents can employ sensory-motor coordination for object detection. In order to obtain a reliable estimate of the situated agent's performance relative to existing object detection methods, we need a larger set of images than is available for the mug-detection task. Therefore we compare it with three different existing object detection methods [9, 10, 16] on the face-detection task for which sufficient images are available.

The remainder of the paper is organised as follows. In Section 2 we describe our situated object detection approach. Then, in Section 3, we describe the experimental setup. In Section 4 we analyse whether and how the evolved situated agents employ sensory-motor coordination. In addition, we address the difference in performance and computational effort between the situated agents and an existing open-loop object detection method [7]. We present a reliable estimate of the performance difference with other methods on the face-detection task in Section 5. In Section 6 we discuss the implication of our experimental results, and in Section 7 we conclude on our research question.

2 Situated Object Detection

In our situated approach to object detection, the detection of an object becomes a process over time that involves a sequence of sensory inputs and motor actions. Before we go into the details regarding the type of sensory inputs and manner of action selection used in our experiments (see Section 3), we give a general overview of the object detection process.

One step of this process is illustrated in Figure 1. At the first time step (t = 1) of a `run', the agent's gaze is initialised at a random location in the

image. The agent takes a local sample from the image by extracting features from a square window centered at the gaze location (indicated by an `x' in Figure 1). Henceforth, we refer to this window as the `fovea'. The extracted features are the sensory inputs of the agent. The agent's controller transforms the extracted features to a motor command, representing a gaze shift in the image (dashed line in the ¯gure). At the new gaze location (`o' in the ¯gure), on t = 2, the agent extracts new features and determines a new gaze shift. The agent performs a sequence of feature extraction and gaze shifting steps until $t = T$. The goal of the agent is to have the gaze location on an object at $t = T$, where T is an experimental parameter (see subsection 3.5).

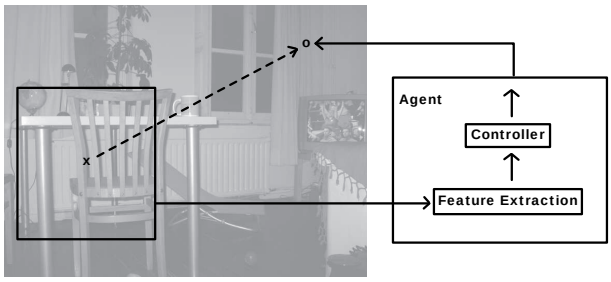

Figure 1: Illustration of a step in the situated object detection process. The agent obtains sensory inputs by extracting features from a square window of gray values in the image. This window, referred to as the fovea, is centered at the gaze location (`x'). The agent's controller maps the sensory inputs to a gaze shift in the image (dashed), determining the next gaze location (`o').

How can the situated agent learn to control its gaze in such a way that it ends up at an object location at $t = T$? One possibility is to use a supervised learning scheme in which the agent learns a mapping from its sensory inputs to an ideal gaze shift that immediately ends up at the object location, as in [17]. However, this training method has the problem that it does not allow an optimal use of sensory-motor coordination, since it excludes non-greedy gaze strategies. Therefore, we opt for a second possibility, namely optimising the agent's controller with an evolutionary algorithm [18]. An evolutionary algorithm enables the optimisation of the agent's controller, taking into account the entire chain of actions and sensory inputs during a run. There is an additional reason why employing an evolutionary algorithm is useful. Namely, if the agent is to ¯nd a class of objects, the sensory inputs should contain some information on the location of such objects. Therefore, the features should capture properties of both the object and its context. Since these properties can vary for di®erent object classes and visual scenes, we also optimise the manner in which features are extracted from the fovea [19]. Employing an evolutionary algorithm allows the simultaneous optimisation of both the agent controller and the feature extraction.

3 Experimental Setup

In this section, we describe the experimental setup. First, we discuss the properties of the agent, its feature extraction and controller (Subsection 3.1). Then, we give details on the evolutionary algorithm (Subsection 3.2) and we provide information on the mug-detection task (Subsection 3.3) and face-detection task (Subsection 3.4). Finally, we provide the experimental parameter settings (Subsection 3.5).

3.1 Agent

In this subsection, we discuss the implementation of the situated object-detection agent for the experiments. First we explain the agent's feature extraction, and then we discuss the agent's controller. Finally, we discuss an extension to the agent, of which preliminary experiments proved the necessity.

For the feature extraction, we adopt the integral features as introduced in [7], because they are e®ective and easy to compute. The evolutionary algorithm can select a speci¯c feature to be extracted from the fovea, by selecting two locations in the fovea to form any rectangular area and selecting any of nine possible input types that are shown in the top part of Figure 2. The value of a feature is the mean gray-value of the image pixels under the white area minus that of the pixels under the grey area. As an illustration, the bottom part of Figure 2 shows a feature of the ¯rst type that covers a large part of the right half of the fovea. The value of this feature is the mean gray-value in image area B minus the mean gray-value in image area A. This feature will respond to vertical contrasts in the right half of the fovea.

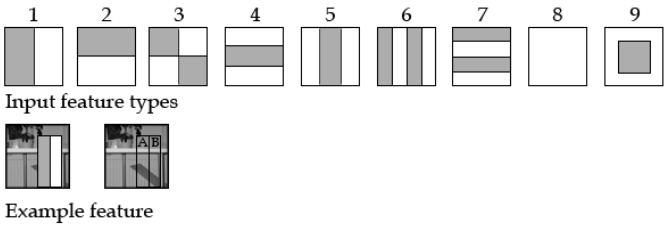

Figure 2: Possible feature types (top part of the ¯gure) and an example feature of type 1 shown in the scanning window (bottom part of the ¯gure). The value of a feature corresponds to the mean gray-value of the image pixels under the white area minus that of the pixels under the grey area.

As the agent's controller, we use a fully connected feedforward neural network. As a result, the agent is reactive: it always selects the same actions for the same sensory inputs [1]. The network has n input neurons that are set to the values of the extracted features, h hidden neurons, and two output neurons. The hidden and output neurons have sigmoid activation functions: $f(x) = \tanh(x)$. There are two output neurons that encode the actions as rela-

tive shifts ($¢x$, $¢y$) in pixels in the image, as follows: $¢x = \lfloor o_1 j \rfloor$, $¢y = \lfloor o_2 j \rfloor$, where j is the maximum number of pixels for a shift.

In our experiments, we employ two agents: one specialised in locations remote from the object and one specialised in locations in the vicinity of the object. The reason for using two agents, rather than one, is twofold. Firstly, at di®erent distances from an object, di®erent visual features are important. Evolving a different agent for distances close to the objects allows the agent to be sensitive to such di®erent visual features. Secondly, previous experiments have shown that evolving agents at di®erent distances and putting them in sequence results in a good detection performance [17]. We put the agents in sequence as follows. The `remote' agent starts at a random location in the image and performs gaze shifts until $t = T$. Then, the `near' agent continues at the end position of the remote agent, performing gaze shifts until $t = 2T$. In each experiment, we evolve the remote agent at uniformly distributed positions in the image and measure its average distance to an object at the end of a run. Then we evolve the near agent at positions close to the objects, according to a normal distribution with its mean on the object center and its standard deviation equal to the average distance to an object reached by the remote agent.

3.2 Evolutionary Algorithm

We employ a `μ, λ' evolutionary algorithm [20] to optimise both the feature extraction and the neural network controller of the agent. Evolution starts with a population consisting of λ individuals. An individual is encoded by its genome, a vector of real values (doubles). Each feature is represented by ¯ve values, one for the type, four for the two coordinates inside the fovea. Each weight of the neural network is encoded by one value. Thus, the genome consists of $5n$ genes for the feature encoding, plus $(h(n+1)+o(h+n+1))$ genes for the neural network (including bias weights). We evaluate each individual by letting it perform R runs per training image, each of T time steps. The ¯tness function we use for each individual i is de¯ned as:

$$f(i) = \text{recall}(i) + (1 - \text{distance}(i)) \quad (1)$$

Where recall(i) is the average proportion of objects detected by the ensemble of R runs. An object is detected if the agent's gaze location is on the object. distance(i) is the average distance to the closest object during a run divided by the maximal distance in the image (taken to be $\sqrt{\text{width}^2 + \text{height}^2}$). The distance-term is included to bootstrap evolution. After evaluating all individuals, we select the μ agents with the highest ¯tness values to form a new generation. The best individual is tested on the validation set of images. This ¯tness value is recorded, so that we can select the best individual of the entire evolution at the end of evolution. Each selected individual has λ/μ o®spring, so that the population size stays the same over time. In producing o®spring, there is a p_c probability that one-point cross-over occurs with one of the other selected individuals, and a $(1-p_c)$ probability that the genome is simply copied.

Afterwards, the genes of the new individual are mutated with a probability of p_m. When a feature gene is mutated, it receives a new random value in the interval [0, 1]. When a weight gene is mutated, the new weight is determined on the basis of the old weight as follows: $w_{new} = w_{old} + r \¢ w$, where r is a random number in the interval $[-1, 1]$ and $\¢ w$ is the change rate. The individuals of the new generation are again evaluated on the images of the training set. The process of ¯tness evaluation and procreation is continued for G generations. Since a change in the features has disruptive e®ects on the usability of the neural controller, we choose to only evolve features in the ¯rst half of evolution. The second half of evolution is dedicated solely to the optimisation of the neural network weights. At generation $G/2$ we change the evolutionary parameters in the following manner. For the optimisation of neural network weights, cross-over might be disruptive [21]. Therefore, we set p_c to 0. In addition, we divide the value of p_m by two at every generation g for which mod$(g, q) = 0$, with q an integer value $\ll G/2$. Finally, we also change the ¯tness function to:

$$f'(i) = \text{recall}(i) \tag{2}$$

in which we discarded the distance term from equation 1. We did this because the distance-term was only meant for bootstrapping evolution.

After evolving the remote agent, we measure its average distance to the closest object. We use this value when we evolve the near agent. The near agent is evolved in the same manner as the remote, except for two di®erences. The ¯rst di®erence is that its initial position in the images is not a random one, but is determined as follows. We randomly select an object in the image and then position the agent at a random location in the vicinity of the object as speci¯ed in subsection 3.1. The second di®erence is that we use an alternative ¯tness function that does not change during evolution:

$$g(i) = \text{precision}(i) + (1 - \text{distance}(i)) \tag{3}$$

The ¯tness function g focuses on precision, the average proportion of the R runs that are located on an object at the end of the run. We include the distance-part, because we want the near agent to approach the nearest object as closely as possible.

At the end of evolution, we select as best agent, the agent that has the highest weighted sum of its ¯tness on the training set and validation set. These ¯tness values are weighted according to the respective sizes of the training and validation sets. This procedure aims to prevent over¯tting to the training set.

3.3 Mug-detection Task

The mug-detection task is a task in which the situated agent has to detect mugs in a domestic environment. The task involves a set of 74 gray-scale images of 1024×980 pixels, containing photos of a domestic environment, where mugs can be located either on the table or in the window. 80% of the image set is used for evolution (60 images) and 20% for testing (14 images). Of the 80% used for

evolution, 80% forms the training set (48 images) and 20% the validation set (12 images). Our results are based on a 5-fold validation scheme.

The mug-detection task is challenging because of the small image set: there is a large variation in object appearance, but there are few examples to learn from (around 80 objects in the set used for evolution). However, the context in which mugs occur is rather constant (for example, they do not °oat in the air). Figure 3 shows some examples of mugs in the dataset (all are image patches of 120×120 pixels). There are many di®erent mugs that are sometimes occluded or rotated (out-of-plane), and there are also some distractor objects that might even fool a human observer, such as the candle in Figure 3 (right object in the third image patch).

Figure 3: Examples of mugs and distractor objects in the mug-detection task.

3.4 Face-detection Task

Although the image set of the mug-detection task is challenging enough, it might be too small to provide a reliable comparison between the situated agent and existing object detection methods. Therefore, we perform this comparison on a real-world face-detection task that has been studied before in [10, 9]. In those studies, three object detection methods were applied to the face-detection task: the Viola and Jones object detector [7], the Fröba-Küllbeck detector [16], and a version of the Viola and Jones detector that always includes a part of the object context in object detection (see [10] and [11]). All three are window-sliding object-detection methods: they check for object presence within a window on all locations of a regular grid on the image. The FGNET-dataset for this task is publicly available (see http://www-prima.inrialpes.fr/FGnet/) and contains images from a video sequence in a meeting room. For our experiments we used the joint set of images from both cameras (`Cam1' and `Cam2') in the ¯rst scene (`ScenA'). We converted the 794 images of 720×576 pixels to gray-scale images. For the FGNET-dataset we use a 2-fold validation scheme. The face-detection task has similar properties to the mug-detection task, although there are more examples to learn from.

3.5 Experimental Settings

In our experiments, we set the number of time steps to T = 5, and the number of runs to R = 20. The agent extracts 10 input features from the fovea. For the neural network, n = 10, h = 5, and o = 2. Its weights are restricted to the interval $[-2, 2]$. Furthermore, the maximal allowed 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 size of the raw input window is one third of the image

width for the remote agent, and one fourth of the image width for the near agent. The evolutionary parameters are as follows: λ = 100, μ = 25, G = 300, p_c = 0.5, p_m = 0.04, ¢w = 0.20, and q = 8. The settings of the experimental parameters are based on preliminary experiments.

4 Mug-detection Task

In this section, we ¯rst analyse the sensory-motor coordination of the agents evolved on the mug-detection task (Subsection 4.1). Then, we turn to their performances (Subsection 4.2).

4.1 Analysis

Our aim is to analyse whether and how the evolved situated agents employ sensory-motor coordination to detect mugs in the images. We perform our analysis on the best agents evolved on the ¯rst training fold, and start with the remote agent, whose gaze is initialised at random locations in the image.

We can answer the question *whether* the situated agent employs sensory-motor coordination by looking at its behaviour. Namely, without sensibly coordinating its sensory inputs and motor actions, a situated agent will not perform better than selecting random locations. Figure 4 shows the behaviour of ten independent runs of the remote agent. Each gaze shift is illustrated with an arrow. The last gaze locations of the runs are marked with a circle. The only mug in the image is indicated with a rectangle. The object in the window in this image is a photoframe. Clearly, the evolved agent employs sensory-motor coordination to some extent. The ¯gure shows that the ensemble of ten runs succeeds in locating the only object in the image (the number of runs is lower than the twenty used during evolution for illustration purposes). Three out of the ten runs succeed in localising the mug in the image, while the other seven runs end up at image locations that are likely to be object locations, i.e., on the table and in the window.

We now proceed by investigating *how* the situated agent coordinates its sensory inputs and motor actions. To this end, we ¯rst analyse the agent's sensory inputs and then its mapping from sensory inputs to motor actions.

The evolved sensory input features capture properties of the object's context. Figure 5 shows the ten evolved input features. The features are projected on an image from the training set and shown at their locations and with their sizes within the fovea (white box). The white cross indicates the center of the fovea. In order to interpret the evolved features, we extracted them at all possible gaze locations within the image and stored their values. We scaled these values to obtain `feature responses' in the interval $[0, 1]$. Figure 6 shows the responses of the di®erent features on all possible gaze locations in the image shown in the bottom right. Light regions mark high responses, dark regions low responses. Figure 5 and 6 illustrate that the agent uses contextual clues to ¯nd the object. This was already suggested by Ballard [14] who stated that

Figure 4: Ten runs of the best evolved remote situated agent of the first fold.

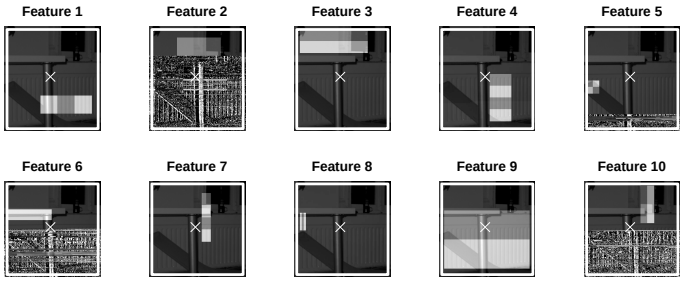

Figure 5: The ten features evolved for the mug-detection task, shown within the scanning window (white box). The white cross indicates the gaze location.

a small object such as a cup can be found by first detecting a larger object such as a table or a lamp and then exploiting its typical spatial relation to the smaller object. It is important to note that the evolved situated agent does not explicitly recognise other objects, since it bases its gaze shifts only on visual input features. However, these features capture properties of large objects that are relevant to object locations. For example, feature 9 (see Figure 5) detects a large horizontal contrast just below the gaze location. This feature is useful in detecting the transition between the floor and the table and wall in the images: its response (Figure 6) forms a gradient from the floor to the table and wall. Feature 2 and 3 seem to be suitable to find the table or window above the gaze location. The response of feature 2 is maximal at the table and the window, while feature 3 has a high response under the table and window. Finding the table or window enhances the agent's chances of subsequently finding the object, since the mugs are usually located on the corresponding flat surfaces. Besides coarse contextual clues, the agent also exploits more detailed features that also seem to be object-related. A nice example is feature 5 (Figure 5) that detects

diagonal contrasts. It has a large response if the gaze location is to the top-right of the lying chair on the °oor (Figure 6). The lying chair is present in many of the images, and is a good indicator of the table position. Note that the evolved situated agent can also detect mugs if the chair is not present (see Figure 8).

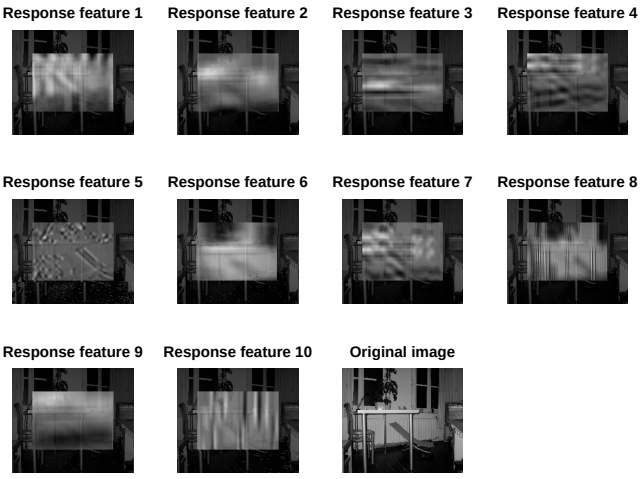

Figure 6: Responses of the ten features shown in Figure 5 in di®erent parts of the image.

To further show that the agent developed input features that carry information on the position of an object, we veri¯ed whether there is a relation between clusters of sensory inputs and the direction and distance to an object. To do this, we gathered local samples in the training set by extracting the evolved features at 30 random gaze locations in each training image (leading to 1440 input samples) and storing the relative distance from the gaze location to the closest object. We clustered the samples with k-means clustering, k = 4. Then, we mapped every sample to the nearest cluster centroid and stored the relative distances for all the samples belonging to a cluster. For each cluster, we can show the relative spatial distribution of inputs belonging to that cluster with respect to the closest object. Figure 7 shows these spatial distributions for the four clusters (the order of the clusters is irrelevant) along with the proportion of sensory inputs belonging to them. The closest object is shown in the middle of every inset with a white cross, and white regions indicate regions where the input cluster has a high probability of occurring (black regions indicate the opposite). The ¯gure shows that the clusters of inputs occur either above or under the closest object at di®erent distances.

The next step in the analysis is to study the agent's mapping from its sensory inputs to its motor actions. The agent exploits the relative spatial distributions of the sensory inputs, but takes into account the entire chain of inputs and actions.

The arrows in Figure 7 represent the direction and size of the gaze shifts

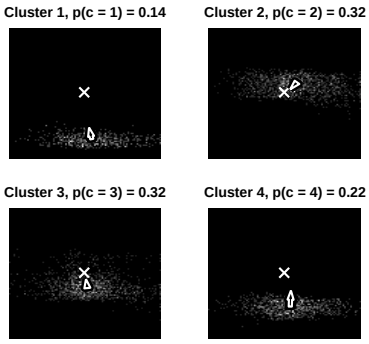

Figure 7: Relative spatial distributions of the four input clusters. White indicates high occurrence, black low occurrence. The cross in the center represents the object location. The arrow is the action taken by the agent, when the input is equal to the cluster centroid. The arrow originates at the median of the relative x- and y- coordinates of the cluster.

taken by the agent when it is provided with the cluster centroids as inputs. The arrows originate from the median relative position of the corresponding cluster of sensory inputs. Clearly, the actions depend on the relative spatial distributions of the sensory input clusters. However, the agent does not seem to apply a greedy action policy: the ideal greedy action would shift the gaze from any position directly to the object position. Therefore, a greedy action policy should at least have resulted in a larger gaze shift for the first input cluster. The smaller shift make more sense if the agent makes more steps to reach an object location. This finding suggests that the agent balances exploiting current information with gathering more information by means of its actions (see [22, 23, 24, 3] for examples of studies on the relation between actions and information gathering).

In spite of the actions shown in Figure 7, one can have doubts whether the agent is very sensitive to the relative spatial distributions of sensory inputs. Namely, the figure also shows that most of the sensory inputs are located below the nearest object. The prior distribution of object locations in the images of the mugs task is such, that mugs mostly occur in the top half of the image (and more often to the left). If the agent were to receive no sensory inputs, its best action would be to move up and slightly to the left. This raises the question whether the agent learns more than the prior distribution of object locations. The agent only receives extracted visual features and does not employ any proprioception with information on its position in the visual scene. It can therefore not exploit the prior distribution of objects directly. However, it can exploit this prior distribution indirectly. If the sensory inputs contain little or no information on the object location, it can select an action that is based on the prior distribution of object locations. The action taken by the agent when provided with the cluster centroid of cluster 2 (Figure 7) shows that the agent

is not insensitive to the spatial relation of the speci¯c input type.

In order to further show that the agent does not just learn a prior distribution, we applied the agent to a manipulated image in which we copy and pasted the table so that its location is lower in the image (opposite to the best a priori action). Figure 8 shows the actions taken by the agent at all points of a grid in the image. It illustrates that in general the agent takes actions towards the table, even if it is located in a direction opposite to the best a priori action. A few of the actions go in the wrong direction (see the top left corner), because of local sampling.

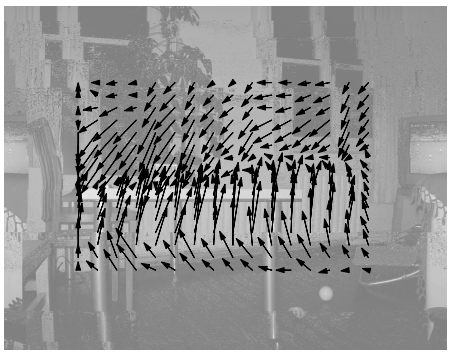

Figure 8: Actions taken by the agent in di®erent parts of the image, for a manipulated image in which the table is located lower in the image.

The analysis of the remote agent clari¯ed in what manner the evolved situated agents employ sensory-motor coordination to detect objects. However, it also showed that the evolved remote agent mainly succeeds in ¯nding the correct y-positions in the image. The near agent, evolved for initial gaze locations that are closer to the objects, exploits object context at a ¯ner scale. It is primarily tuned to properties of the object itself and is useful for determining the exact location of the nearest object. Figure 9 shows the actions taken by the near agent in the vicinity of an object. From this ¯gure it is clear that the agent can approach objects on a ¯ner scale.

Figure 9: Actions taken by the near agent for gaze locations near the object.

Table 1: Performance on the mug-detection task.

	recall in % (σ)	false positives / image (σ)
remote agent	56.7 (14)	18.4 (0.5)
sequential agent	68.1 (6)	16.7 (0.5)
Viola and Jones	34.4	15.0
sequential agent + VJ	31.9	2.0

4.2 Performance

We now turn to the quantitative performance of the evolved agents. Table 1 shows the performance in terms of recall (proportion of objects that are detected) and number of false positives per image on the mug-detection task. A false positive is a location that is mistakenly considered as an object location. The results are averaged over all ¯ve experiments of the folded test, with standard deviations between brackets. The remote agent alone achieves an average recall of 56.7% at the cost of 18.4 false positives and the sequential agent (remote and near agent together) achieves a recall of 68.1%, at the cost of 16.7 false positives. In order to evaluate this performance in relation to a state-of-the-art object detector, we trained a Viola and Jones-classi¯er for the ¯rst fold of the mugs task. We employed the supervised training scheme explained in [7] that trains the classi¯er to discern between samples containing the object or containing the background. The classi¯er evaluates object presence at all points of a regular grid in the image on the basis of the same type of features as used by our approach. The performance of this classi¯er is shown in the third row of the table. At an approximate equal number of false positives as the situated agent, it detects only half as many objects. The situated object-detection agent compares favourably to the Viola and Jones object detector. However, the number of false positives is still very high: an average of 16.7 false positives per image as compared to an average of roughly two objects per image. The main problem that remains for the situated agent is that it cannot recover from local minima that do not contain an object. Therefore, we extend it by applying the Viola and Jones classi¯er at the end of a run. The results of this extension are shown in the fourth row of the table. Recall is almost as high as that of the Viola and Jones classi¯er alone, but the average number of false positives is much lower.

Importantly, this performance is obtained while extracting far fewer local samples. The situated agent uses $R(2T)$ local samples, where R is the number of runs and T the number of time steps per type of agent (remote / near). $R(2T-1)$ of these samples are used for reaching the ¯nal gaze location and R for verifying the presence of objects with the Viola and Jones classi¯er. In our experiments R = 20 and T = 5, so 200 local samples are used. The Viola and Jones classi¯er checks object presence at all points of a $G_1 \times G_2$ grid, resulting in the use of $G_1 G_2$ local samples. In our experiments G_1 = G_2 = 100, so

10,000 samples are used. While using 50 times fewer samples, the situated object-detection approach still outperforms the Viola and Jones classifier on the mug-detection task. The computational effort saved is considerable, but depends on the type of features that are extracted.

5 Face-detection Task

Analysis of the agents evolved for the face-detection task gives results similar to those for the mug-detection task. However, the larger image set of the face-detection task allows more reliable performance estimates. In this section, we therefore focus on comparing the situated agent's performance with that of other methods.

In the field of object detection, it is common to illustrate detection performance with an FROC-plot. This plot shows the trade-off between recall and the average number of false positives per image. Since most object-detection methods use a binary classifier in some stage of the object-detection process, such an FROC-curve can be constructed by varying the threshold of the classifier. A lower threshold implies more detections, but also more false positives. A higher threshold implies fewer false positives, but also fewer detections. Because the situated object-detection agent does not always use a binary classifier, it is less obvious how we should construct the FROC-curve. Although we have multiple options for generating an FROC-curve (e.g., changing the fitness functions), we choose to generate the FROC-curve by varying the number of independent runs R on the test images ($R \in \{1, 3, 5, 10, 20, 30\}$). More runs will result in higher recall and more false positives, while less runs will result in lower recall and fewer false positives. Figure 10 shows the FROC-curves of the different versions of the situated agents (square markers) and of the three open-loop object detection methods studied in [10, 9]: the Viola and Jones object detector [7] (`o'-markers), the Fröba and Küllbeck-detector [16] (`+'-markers), and the detector introduced in [10] (`$\times$'-markers). The experimental results from [9] are shown with thin lines, the results from [10] with thick lines.

A better object detector will have an FROC-curve that is more to the top left of the figure: it achieves high recall with few false positives. Figure 10 shows that the situated object-detection agent outperforms the naive application of the Viola and Jones classifiers and the Fröba-Küllbeck classifier, while performing slightly worse than the approach of Kruppa et al. The main reason for the fact that the Viola and Jones classifier and Fröba-Küllbeck classifier are outperformed by both the situated agents and the method of Kruppa et al., is that the latter two methods exploit context in their object detection. Although the faces in the FGNET data set can have very different appearances, there is a relatively fixed context. For example, Kruppa's method exploits the presence of shoulders below the face as an indication for the presence of a face. The evolved situated agents additionally exploit properties of the meeting room, such as the fact that the walls are relatively homogeneous (as they often are in office environments). The agents follow the wall downwards, while being attracted by

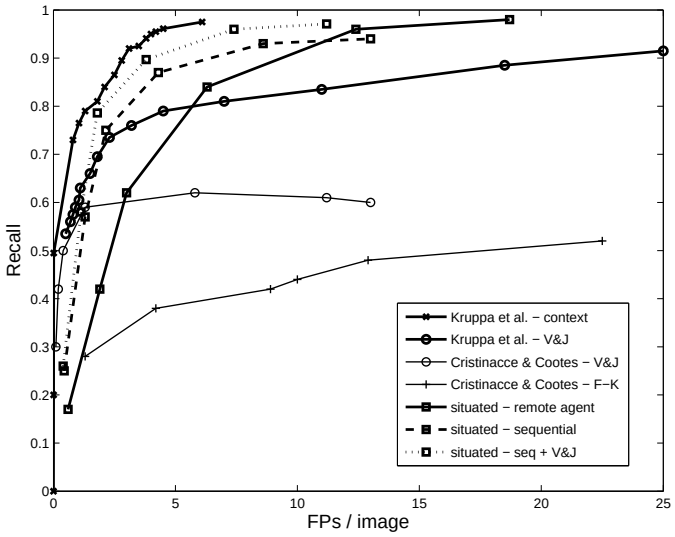

Figure 10: FROC-curve for the results on the FGNET face-detection task from different studies.

head-like shapes. The exploitation of context leads to a higher robustness in object detection (this was also observed in [10, 11]).

Interestingly, the Viola and Jones classifier achieves different performances for the studies [10] and [9]. There are several possible explanations for this, since there are slight differences between the studies (slightly different sets selected from the FGNET video sequence, labelings, and parameter settings). However, the difference that might best explain the difference is that in [10] the training set consists of images that are not part of the FGNET data set. This implies that it is difficult to very accurately compare the results from the different studies. However, they are reliable enough to conclude that the situated agents perform at par with state-of-the-art object detection methods on the FGNET face-detection task.

6 Discussion

In this section, we first discuss the disadvantage of the situated approach to object detection. Then, we discuss the experimental results in light of the criticism that ECS tackles simple tasks.

The advantages of the situated object detection approach (robust detection and computational efficiency) come at the expense of application generality. Since the situated agents exploit visual properties of the object's context in a visual scene, it becomes dependent on these properties for object detection. Therefore, we cannot expect a situated agent evolved on the FGNET data set to detect faces in random images from the web. The extent to which application

generality is limited is an issue that deserves further research.

In the introduction we motivated our research on situated object detection by referring to the criticism that ECS only concerns toy tasks. Beer already responded to this criticism by arguing that minimally cognitive agents (that tackle simple tasks) could be as insightful for cognitive science as frictionless models have been for physics [25]. Therefore, one could wonder whether the empirical results reported in Section 4 and 5 help in addressing the criticism. We believe they do, for the following two reasons. First, the task of object detection in static natural images cannot be argued to be a toy task, since existing object detection methods still are not successful compared to human performance. Scaling ECS up to object detection in static natural images might perhaps not add much to the functioning of situated agents as frictionless models, but at least it takes away this criticism on the approach. Second, the empirical results show that situated object detection performs at par with state-of-the-art object-detection methods on the FGNET dataset, while being computationally more e±cient. The presented results are not yet convincing enough so that engineers will apply situated object detection instead of more classical methods, but the results illustrate the potential of situated object detection to become an engineering tool for object detection. We consider it important to continue developing situated object detection, because this e®ort can provide us with empirical evidence for the advantages of situatedness in visual tasks such as object detection. In addition, it can provide us with concrete examples of how sensory-motor coordination can play a role in object detection tasks that humans face as well.

7 Conclusion

We conclude by stating that the studied (arti¯cial) situated agents can employ sensory-motor coordination to successfully perform object detection in static natural images. The results show that sensory-motor coordination allows for robust object detection, because of the exploitation of an object's visual context. The situated agents perform at par with existing window-sliding object-detection methods, while being computationally more e±cient. This computational e±ciency results from the fact that sensory-motor coordination allows e±cient selective extraction of local image samples.

Acknowledgment

We would like to thank Ben Torben-Nielsen for commenting on an earlier version of this paper. This research was carried out with funding of the Limburg / UM SWOL fund, and funding of the Netherlands Organisation for Scienti¯c Research (NWO), under grant number 634.000.018.

References

[1] S. Nol¯, \Power and the limits of reactive agents," *Neurocomputing*, vol. 42, pp. 119{145, 2002.

[2] M. F. van Dartel, I. G. Sprinkhuizen-Kuyper, E. O. Postma, and H. J. van den Herik, \Reactive agents and perceptual ambiguity," *Adaptive Behavior*, vol. 13, no. 3, pp. 227{242, 2005.

[3] G. de Croon, E. O. Postma, and H. J. van den Herik, \A situated model for sensory-motor coordination in gaze control," *Pattern Recognition Letters*, vol. 27, pp. 1181{1190, 2006.

[4] J. Teo and H. Abbass, \Multiobjectivity and complexity in embodied cognition," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 4, pp. 337{360, 2005.

[5] R. D. Beer, \The dynamics of active categorical perception in an evolved model agent," *Adaptive Behavior*, vol. 11:4, pp. 209{243, 2003.

[6] S. Edelman, \But will it scale up? not without representations," *Adaptive Behavior*, vol. 11, pp. 273{275, 2003.

[7] P. Viola and M. J. Jones, \Robust real-time object detection," *Cambridge Research Laboratory, Technical Report Series*, 2001.

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

[9] D. Cristinacce and T. Cootes, \A comparison of two real-time face detection methods," in 4^{th} *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2003, pp. 1{8.

[10] H. Kruppa, M. Castrillon-Santana, and B. Schiele, \Fast and robust face ¯nding via local context," in *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS'03), Nice, France*, 2003.

[11] N. H. Bergboer, E. O. Postma, and H. J. van den Herik, \A context-based model of attention," in *Proceedings of the 16th European Conference on Arti¯cial Intelligence (ECAI), Valencia, Spain*, 2004, pp. 927{931.

[12] J. K. O'Regan and A. Noë, \A sensorimotor account of vision and visual consciousness," *Behavioral and Brain Sciences*, vol. 24:5, pp. 883{917, 2001.

[13] D. Floreano, T. Kato, D. Marocco, and E. Sauser, \Coevolution of active vision and feature selection," *Biological Cybernetics*, vol. 90:3, pp. 218{228, 2004.

[14] D. H. Ballard, \Animate vision," *Arti¯cial Intelligence*, vol. 48, pp. 57{86, 1991.

[15] J. Schmidhuber and R. Huber, \Learning to generate arti¯cial fovea trajectories for target detection," *International Journal of Neural Systems*, vol. 2, pp. 135{141, 1991.

[16] B. Fröba and C. Küllbeck, \Robust face detection at video frame rate based on edge orientation features," in 5^{th} *international conference on automatic face and gesture recognition 2002*, 2002, pp. 342{347.

[17] G. de Croon and E. O. Postma, \Active object detection," in *Belgian-Dutch AI Conference, BNAIC 2006, Namur, Belgium*, 2006.

[18] S. Nol¯ and D. Floreano, *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA, MIT Press/Bradford Books, 2000.

[19] I. Macinnes and E. Di Paolo, \The advantages of evolving perceptual cues," *Adaptive Behavior*, vol. 14, no. 2, pp. 147{156, 2006.

[20] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, Oxford, 1996.

[21] X. Yao, \Evolving arti¯cial neural networks," *Proceedings of the IEEE*, vol. 87, pp. 1423 { 1447, 1999.

[22] D. Kirsh and P. Maglio, \On distinguishing epistemic from pragmatic action," *Cognitive Science*, vol. 18, pp. 513{549, 1994.

[23] A. Klyubin, D. Polani, and C. Nehaniv, \Organization of the information °ow in the perception-action loop of evolved agents," in *Proceedings of 2004 NASA/DoD Conference on Evolvable Hardware*, R. Zebulum, D. Gwaltney, G. Hornby, D. Keymeulen, J. Lohn, and A. Stoica, Eds. IEEE Computer Society, 2004, pp. 177{180.

[24] M. Lungarella, T. Pegors, D. Bulwinkle, and O. Sporns, \Methods for quantifying the information structure of sensory and motor data," *Neuroinformatics*, vol. 3, no. 3, p. 243262, 2005.

[25] R. D. Beer, \Arches and stones in cognitive architecture: Reply to comments," *Adaptive Behavior*, vol. 11, pp. 299{305, 2003.