Selecting what is Important: Training Visual Attention

Simone Frintrop[1], Gerriet Backer[2], and Erich Rome[1]

[1]Fraunhofer Institut für Autonome Intelligente Systeme (AIS),
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
[2]Krauss Software GmbH, Tiefe Str. 1, 38162 Cremlingen, Germany

Abstract. We present a new, sophisticated algorithm to select suitable training images for our biologically motivated attention system VOCUS. The system detects regions of interest depending on bottom-up (scene-dependent) and top-down (target-specific) cues. The top-down cues are learned by VOCUS from one or several training images. We show that our algorithm chooses a subset of the training set that outperforms both the selection of one single image as well as simply using all available images for learning. With this algorithm, VOCUS is able to quickly and robustly detect targets in numerous real-world scenes.

1 Introduction and State of the Art

Human visual perception is based on a separation of object recognition into two subtasks [11]: first, a fast parallel pre-selection of scene regions detects object candidates and second, complex recognition restricted to these regions verifies or falsifies the hypothesis. This dichotomy of fast localization processes and complex, robust, but slow identification processes is highly effective: expensive resources are guided towards the most promising and relevant candidates.

In computer vision, the efficient use of resources is equally important. Although an attention system generates a certain overhead in computation, it pays off since reliable object recognition is a complex vision task that is usually computationally expensive. The more general the recognizer – for different shapes, poses, scales, and illuminations – the more important is a pre-selection of regions of interest.

Concerning visual attention, research has so far been focused on localizing the relevant scene parts by evaluating bottom-up, data-driven saliency, featuring research from a psychological [16, 19], neuro-biological [2, 12] and computational [8, 7, 1, 14] point of view. Koch & Ullman [8] described the first explicit computational architecture for bottom-up visual attention. It is strongly influenced by Treisman's *feature-integration theory* [16] and already contains the main properties of many current visual attention systems, e.g., the one by Itti et al. [7] or [1, 14]. These systems use classical linear filter operations for feature extraction, rendering them especially useful for real-world scenes. Another approach is provided by models consisting of a pyramidal neural processing architecture, e.g., the *selective tuning model* by Tsotsos et al. [18].

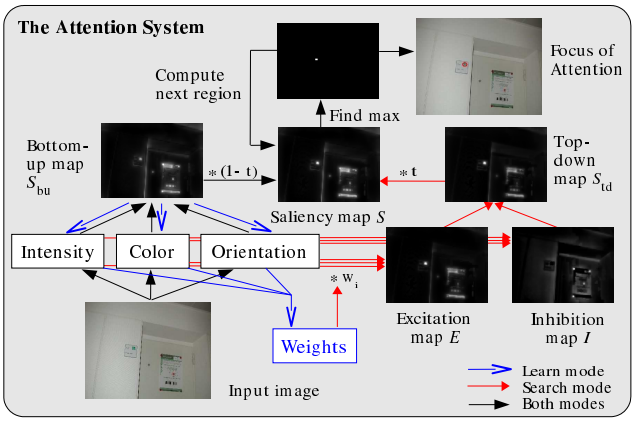
While much less analyzed, there is strong evidence for top-down influences modifying early processing of visual features due to motivations, emotions, and goals [2]. Only a few computer models of visual attention integrate top-down information into their systems. The earliest approach is the *guided search* model by Wolfe [19], a result of his psychological investigations of human visual search performance. Tsotsos's strongly biologically motivated system considers feature channels separately and uses inhibition for regions of a specified location or those that do not fit the target features [18]. Schill et al. [13] use top-down information from a knowledge-base to select actual fixations from the fixation candidates determined by a bottom-up system. The computation of the bottom-up features is not influenced by the top-down information. Other systems enabling goal-directed search are presented by Hamker [5] and by Navalpakkam et al. [10]; however, Hamker's system does not consider the surroundings of the targets and both do not separate enhancing and inhibiting cues as well as bottom-up and top-down cues. We are not aware of any other well investigated system of top-down visual attention comparable to our approach including the automated learning of features and considering features of both the target and the surroundings.

Our attention system VOCUS was introduced in [4, 3], and applied to object detection and recognition in [9, 3]. VOCUS performs goal-directed search by extending a well-known bottom-up attention architecture [7] by a top-down part. The bottom-up part computes saliencies in the feature dimensions intensity, orientation, and color independently, weights maps according to the exclusivity of the feature, and finally fuses the saliencies into a single map. The top-down part uses in a search phase learned feature weights to determine which features to enhance and which ones to inhibit. The weighted features contribute to a top-down saliency map highlighting regions with target-relevant features. This map is integrated with the bottom-up map, resulting in a global saliency map and the focus of attention is directed to its most salient region.

The focus of this paper is the learning phase, where relevant feature values for a target are learned using one or several training images. Here, the system automatically determines which features to regard in order to separate the target best from its surroundings.

Learning weights from one single training image yields good results when the target object occurs in all test images in a similar way, i.e., the background color is similar and the object occurs always in a similar orientation. But when the targets occur on different backgrounds and are rendered from different viewpoints, several training images have to be considered for learning. This is important for us as we intend to integrate the system into a robot control architecture enabling the detection of salient regions and goal-directed search in dynamic environments.

So we devised a new, sophisticated algorithm that chooses suitable training images from a training set. It is shown that training on the chosen subset yields much better results than training on a single or on more training images: the subset is a local optimum. Search results for various real-world scenes are presented, showing that VOCUS is robust and applicable to natural scenes.



Feature	weights
intensity on/off	0.001
intensity off/on	9.616
orientation 0°	4.839
orientation 45°	9.226
orientation 90°	2.986
orientation 135°	8.374
color green	76.572
color blue	4.709
color red	0.009
color yellow	0.040
conspicuity I	6.038
conspicuity O	5.350
conspicuity C	12.312

Fig. 1. The goal-directed visual attention system with a bottom-up part (left) and a top-down part (right). In learning mode, target weights are learned (blue arrows). These are used in search mode (red arrows). Right: weights for target name plate.

In section 2, we start by explaining the attention system VOCUS. Section 3 extends the learning of target-specific weights to several training images and introduces our new algorithm that chooses the most suitable images from a training set. In section 4, we present numerous results on real-world images before we conclude in section 5.

2 The Attention System VOCUS

In this section, we present the goal-directed visual attention system VOCUS (Visual Object detection with a CompUtational attention System) (cf. Fig. 1). With visual attention we mean a selective search-optimization mechanism that tunes the visual processing machinery to approach an optimal configuration [17]. VOCUS consists of a bottom-up part computing data-driven saliency and a top-down part enabling goal-directed search. The global saliency is determined from bottom-up and top-down cues. More details to VOCUS are found in [3].

2.1 Bottom-up saliency

Feature Computations: The first step for computing bottom-up saliency is to generate image pyramids for each feature to enable computations on different scales. Three features are considered: Intensity, orientation, and color. For the feature intensity, we convert the input image into gray-scale and generate a Gaussian pyramid with 5 scales s_0 to s_4 by successively low-pass filtering and subsampling the input image, i.e., $s_{(i+1)}$ has half the width and height of s_i.
The intensity maps are created by center-surround mechanisms, which compute the intensity differences between image regions and their surroundings. We compute two kinds of maps, the on-center maps I''_{on} for bright regions on dark background, and the off-center maps I''_{off}: Each pixel in these maps is computed

by the difference between a center c and a surround σ (I''_{on}) or vice versa (I''_{off}). Here, c is a pixel in one of the scales s_2 to s_4, σ is the average of the surrounding pixels for two different radii. This yields 12 intensity scale maps $I''_{i,s,\sigma}$ with $i \epsilon \{\text{on}, \text{off}\}, s \epsilon \{s_2\text{-}s_4\}$, and $\sigma \epsilon \{3, 7\}$. The maps for each i are summed up by *inter-scale addition* $\bigoplus$, i.e., all maps are resized to scale 2 and then added up pixel by pixel yielding the intensity feature maps $I'_i = \bigoplus_{s,\sigma} I''_{i,s,\sigma}$.

To obtain the orientation maps, four oriented Gabor pyramids are created, detecting bar-like features of the orientations $\theta = \{0\,^\circ, 45\,^\circ, 90\,^\circ, 135\,^\circ\}$. The maps 2 to 4 of each pyramid are summed up by inter-scale addition yielding 4 orientation feature maps O'_θ.

To compute the color feature maps, the color image is first converted into the uniform CIE LAB color space [6]. It represents colors similar to human perception. The three parameters in the model represent the luminance of the color (L), its position between red and green (A) and its position between yellow and blue (B). From the LAB image, a color image pyramid P_{LAB} is generated, from which four color pyramids P_R, P_G, P_B, and P_Y are computed for the colors red, green, blue, and yellow. The maps of these pyramids show to which degree a color is represented in an image, i.e., the maps in P_R show the brightest values at red regions and the darkest values at green regions. Luminance is already considered in the intensity maps, so we ignore this channel here. The pixel value $P_{R,s}(x, y)$ in map s of pyramid P_R is obtained by the distance between the corresponding pixel $P_{\text{LAB}}(x, y)$ and the prototype for red $r = (r_a, r_b) = (255, 127)$. Since $P_{\text{LAB}}(x, y)$ is of the form (p_a, p_b), this yields: $P_{R,s}(x, y) = ||(p_a, p_b), (r_a, r_b)|| = \sqrt{(p_a - r_a)^2 + (p_b - r_b)^2}$.
On these pyramids, the color contrast is computed by on-center-off-surround differences yielding 24 color scale maps $C''_{\gamma,s,\sigma}$ with $\gamma \epsilon \{\text{red}, \text{green}, \text{blue}, \text{yellow}\}, s \epsilon \{s_2\text{-}s_4\}$, and $\sigma \epsilon \{3, 7\}$. The maps of each color are inter-scale added into 4 color feature maps $C'_\gamma = \bigoplus_{s,\sigma} \hat{C}_{\gamma,s,\sigma}$.

Fusing Saliencies: All feature maps of one feature are combined into a conspicuity map yielding one map for each feature: $I = \sum_i \mathcal{W}(I'_i)$, $O = \sum_\theta \mathcal{W}(O'_\theta)$, $C = \sum_\gamma \mathcal{W}(C'_\gamma)$. The bottom-up saliency map S_{bu} is finally determined by fusing the conspicuity maps: $S_{bu} = \mathcal{W}(I) + \mathcal{W}(O) + \mathcal{W}(C)$

The exclusivity weighting $\mathcal{W}$ is a very important strategy since it enables the increase of the impact of relevant maps. Otherwise, a region peaking out in a single feature would be lost in the bulk of maps and no pop-out would be possible. In our context, important maps are those that have few highly salient peaks. For weighting maps according to the number of peaks, each map X is divided by the square root of the number of local maxima m that exceed a threshold t: $\mathcal{W}(X) = X/\sqrt{m} \quad \forall m : m > t$. Furthermore, the maps are normalized after summation relative to the largest value within the summed maps. This yields advantages over the normalization relative to a fixed value (details in [3]).

The Focus of Attention (FOA): To determine the most salient location in S_{bu}, the point of maximal activation is located. Starting from this point, region

growing recursively finds all neighbors with similar values within a threshold and the FOA is directed to this region. Finally, the salient region is inhibited in the saliency map by zeroing, enabling the computation of the next FOA.

2.2 Top-down saliency

Learning mode: In learning mode, VOCUS is provided with a training image and coordinates of a rectangle depicting the *region of interest (ROI)* that includes the target. Then, the system computes the bottom-up saliency map and the *most salient region (MSR)* inside the ROI. So, VOCUS is able to decide autonomously what is important in a ROI, concentrating on parts that are most salient and disregarding the background or less salient parts.

Next, weights are determined for the feature and conspicuity maps, indicating how important a feature is for the target (blue arrows in Fig. 1). The weight w_i for map i is the ratio of the mean saliency in the target region $m_{(MSR)}$ and in the background $m_{(image-MSR)}$: $w_i = m_{(MSR)}/m_{(image-MSR)}$. This computation does not only consider which features are the strongest in the target region, it also regards which features separate the region best from the rest of the image. In section 3, we show how the approach is extended to several training images.

Search mode: In search mode, firstly the bottom-up saliency map is computed. Additionally, we determine a top-down saliency map that competes with the bottom-up map for saliency (red arrows in Fig. 1). The top-down map is composed of an excitation and an inhibition map. The excitation map E is the weighted sum of all feature and conspicuity maps X_i that are important for the learned region, i.e., $w_i > 1$. The inhibition map I shows the features more present in the background than in the target region, i.e., $w_i < 1$:

$$\begin{array}{ll} E = \sum_i (w_i * X_i) & \forall i : w_i > 1 \\ I = \sum_i ((1/w_i) * X_i) & \forall i : w_i < 1 \end{array} \quad (1)$$

The top-down saliency map $S_{(td)}$ is obtained by: $S_{(td)} = E - I$. The final saliency map S is composed as a combination of bottom-up and top-down influences. When fusing the maps, it is possible to determine the degree to which each map contributes by weighting the maps with a top-down factor $t \in [0..1]$: $S = (1-t) * S_{(bu)} + t * S_{(td)}$.

With $t = 1$, VOCUS looks only for the specified target. With $t < 1$, also bottom-up cues have an influence and may divert the focus of attention. This is also an important mechanism in human visual attention: a person suddenly entering a room or a deer jumping on the road catch immediately our attention, independently of the task. In humans this *attentional capture* cannot be overridden by top-down search strategies [15]; i.e., for a severely biologically motivated system a top-down factor of 1 should not be allowed. Nevertheless, for a technical system that usually has to solve only one clearly defined task at a time, also a top-down factor of 1 is often useful.

The success of the search is evaluated by the rank of the focus that hits the target, denoted by the **hit number**. For example, if the 2nd focus is on the

target, the hit number is 2. The lower the hit number, the better the search performance. If the hit number is 1, the target is immediately detected. In pop-out experiments, the hit number is 1 by definition. If a whole image set is evaluated, we determine the **average hit number**, i.e., the arithmetic mean of the hit numbers of all images. Usually, only a determinate number of fixations is considered so that images with undetected targets are not included in the average. To indicate this, we show in our experimental results the percentage of detected targets (**detection rate**) additionally.

3 Using Several Training Images

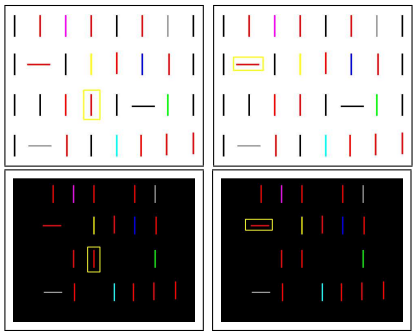
Learning weights from one single training image yields good results when the target object occurs in all test images in a similar way, i.e., the background color is similar and the object occurs in nearly identical orientations. These conditions often occur if the objects are fixed elements of the environment. For example, name plates or fire extinguishers usually are placed on the same kind of wall, so the background has always a similar color and intensity. Furthermore, since the object is fixed, its orientation does not vary and it is sensible to learn that fire extinguishers usually have a vertical orientation.

Although the search is already quite successful with weights from a single training image, the results differ somewhat depending on the choice of the training image. This is shown in Tab. 1: a highlighter was searched in a test set of 60 images using the weights from a single training image. The table shows the different results for several training images; the detection rate differs between 95 and 100%.

Furthermore, the results usually differ slightly depending on the test set the weights are applied to. One training image might fit better to a special image set than to another. To weed out these special cases, it is sensible to take the average weight of at least two training images to enable a more stable performance on arbitrary test sets. For movable objects it is even more important to compute average weights. A highlighter may lie on a dark or on a bright desk and it may have any orientation. Here, it is necessary to learn from several training images which features are stable and which are not.

Table 1. The search for a highligher with the weights w_1 to w_5 learned from 5 training images applied to a test set of 60 images (examples of training and test images in Fig. 3 and 4). The first 10 foci were determined. The performance is shown as the average hit number and the percentage of targets detected within the first 10 foci in parentheses. The performance differs slightly depending on the training image.

	# test	average hit number (and detection rate [%])				
Target	im	w_1	w_2	w_3	w_4	w_5
Highlighter	60	1.83 (99%)	1.70 (97%)	1.43 (100%)	1.93 (95%)	1.78 (97%)



Feature	weights for red bar v,b	h,b	v,d	h,d	average
int on/off	0.00	0.01	**8.34**	**9.71**	0.14
int off/on	**14.08**	**10.56**	0.01	0.04	0.42
ori 0°	1.53	**21.43**	0.49	**10.52**	3.61
ori 45°	2.66	1.89	1.99	2.10	2.14
ori 90°	**6.62**	0.36	**5.82**	0.32	1.45
ori 135°	2.66	1.89	1.99	2.10	2.14
col green	0.00	0.00	0.00	0.00	0.00
col blue	0.00	0.00	0.01	0.01	0.00
col red	**18.87**	**17.01**	**24.13**	**24.56**	**20.88**
col yellow	**16.95**	**14.87**	**21.21**	**21.66**	**18.45**
consp I	7.45	5.56	3.93	4.59	5.23
consp O	4.34	7.99	2.87	5.25	4.78
consp C	4.58	4.08	5.74	5.84	5.00

Table 2. Left: four training examples to learn red bars of horizontal and vertical orientation and on different backgrounds. The target is marked by the yellow rectangle. Right: The learned weights. Column 2–5: the weights for a single training image (vertical bar on bright background (v,b), horizontal on bright (h,b), vertical on dark (v,d), horizontal on dark (h,d)). The highest values are highlighted in bold face. Column 6: average weights. Color is the only stable feature.

3.1 Average Weights

To achieve a robust target detection even in changing environments, it is necessary to learn the target properties from several training images. This is done by computing the average weight vector from n training images with the *geometric mean* of the weights for each feature, i.e., the average weight vector $\boldsymbol{w}_{(1,..,n)}$ from n training images is determined by:

$$\boldsymbol{w}_{(1,..,n)} = \sqrt[n]{\prod_{j=1}^{n} \boldsymbol{w}_j}. \tag{2}$$

The geometric mean is more suitable than an arithmetic mean, because the weight values represent relations, so that values like 2 and 0.5 should cancel out each other. If one feature is present in some training images but absent in others, the average values will be close to 1 leading to only a low activation in the top-down map. In Tab. 2 this is shown on the example of searching for red bars: the target occurs in horizontal or vertical orientations and on a dark or bright background; the only stable feature is the red color. This is reflected in the rightmost column of the table which shows the average weights: the weights for intensity and orientation feature maps are almost equal, only weights for the color feature maps show high values. This enables the search for red bars, regardless of the background and the orientation.

3.2 The Algorithm to Choose the Training Images

In the previous example (Tab. 2), four training images were chosen that were claimed to represent the test data. In practice, the problem is: how do we find suitable training images? We could think about the test application and reason about suitable training data or we could just use a bunch of training images that cover many possible contexts presenting the target at different orientations and on different backgrounds. However, this does not guarantee a good training set and, moreover, it depends heavily on the user's experiences and skills. In this section, we introduce an algorithm that chooses the most suitable images from a training set.

Let us first think about how an optimal weight vector could be achieved. Since the average weights do not always improve when more training images are considered, the best performance is usually not achieved by considering all images of a training set T_1. The reason is that training on too similar images results in *overfitting*, e.g., generating too specialized weights. Instead, there exists a subset of T_1, the average weights of which yield the best performance on another image set T_2. The only possibility to find this subset is to test all possible combinations, an effort costing to check 2^n combinations for n training images. Since these computations are too costly even for rather small n, we propose an approximation algorithm that yields a local optimum in performance. Before we introduce the algorithm, we first give a definition:

Definition 1 (Self-test, self-test hit number). *A* **self-test** *on image I for a target t means: first, learn the weights $\boldsymbol{w}$ for t from image I. Second, apply $\boldsymbol{w}$ to I itself. The resulting hit number is the* **self-test hit number.**

The self-test hit number is a good base for comparisons, since the weights of an image itself yield a good chance to discriminate the target from its surrounding. A self-test hit number of 1 indicates that the weights are sufficient to detect the target in similar environments. A larger self-test hit number indicates that there are distractors in the scene that are very similar to the target and that the features of the system are incapable to distinguish between target and distractors. This test is not suitable for deciding whether a training image is useful or not, because if there are distractors in a scene which are too similar according to the given features, there is nothing we can do about it. It might be useful to train on such scenes anyway, because the extracted weights are the best possible solution for these kinds of scenes. Note that a hit number of 2, 3, or even 10 is often still useful since the regions to be investigated by an object classifier are still considerably reduced.

The overall idea of the approximation training algorithm is to first choose one arbitrary image I_1 from the training image set. Then, the weights from I_1 are applied to the whole training set T and the image I_2 is determined on which the hit number is worst. A bad hit number might mean that I_1 was not suitable for this image. Whether this assumption is true can be checked by comparing the hit number with the self-test hit number of I_2. If the latter is better, the assumption was true: I_1 was unsuitable. In this case, I_2 is a good choice to improve the

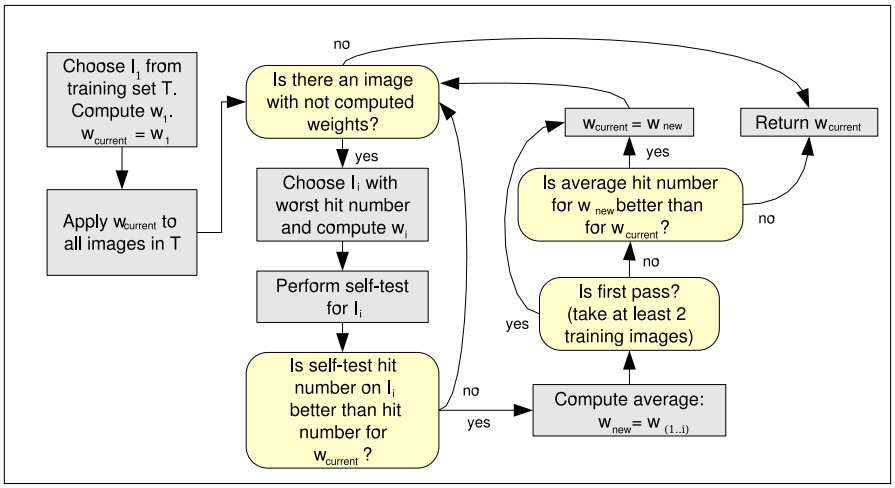


Fig. 2. The algorithm to find the most suitable training images out of an image set and to compute their average weight vector. With this vector, a local optimum in detection quality on the training set is achieved.

weights thus the average weights of I_1 and I_2 are determined. This procedure is continued as long as the average hit rate on the training set improves. A flowchart of the algorithm is shown in Fig. 2.

4 Results

In this section, we show the hit numbers and detection rates of VOCUS when searching for several targets in various real-world scenes. We compare the performance for one and for several training images which were chosen with the presented algorithm. As targets, we used four kinds of objects: two objects which are fixed in our office environment (fire extinguishers and name plates) and two movable objects (a key fob and a highlighter). The highlighter was presented on two different desks, a dark (black) and a bright (wooden) one. For each target, we used a training set of 10 to 54 images and chose suitable training images from the set with the training algorithm in Fig. 2. In Fig. 3, we depict one of the training images for each target as well as the most salient region that VOCUS extracted for learning.

Two experiments demonstrate the power of our algorithm: first, we examine the search performance on four test sets using different numbers of training images. Second, we test VOCUS on a search task in which the environment of the target differs and show that in this case more training images are required.

Fig. 3. Top: the training images with the targets (name plate, fire extinguisher, key fob, and a highlighter on the dark desk). Bottom: The part of the image that was marked for learning (region of interest (ROI)) and the contour of the region that was extracted for learning (most salient region (MSR)).

Experiment 1 Our first experiment examines the effect of different image set sizes on the search quality. We show the search results with the computed vectors first on the training set itself (training phase) and then on a test set (test phase).

The image sets used for this experiment consisted of images with similar backgrounds for each set (white walls behind the fire extinguisher and the name plate and same desk for the key fob and the highlighter). The images were nevertheless highly complex and include a heavily structured surrounding with many distracting regions.

To compute the weight vectors, we chose the most suitable training images from the training set using the algorithm of Fig. 2. Remember that the algorithm chooses the first image at random and then the images with the worst detection results on the training set; it stops when a local optimum in performance is reached. We document this by presenting the detection results (average hit number and detection rate) for each weight vector that is computed during the iterations of the algorithm (Tab. 3). This corresponds to visualizing the intermediate steps of the algorithm.

It turned out that averaging two training images in most cases outperformed the use of a single image. This is most obvious for the name plate: the detection rate within the first 10 foci increased from 87% to 94% (the detection rate is more important to evaluate the performance than the average focus because a single image that is additionally detected increases the detection rate slightly but decreases the average hit number. That means, a performance of average hit number 2.04 and detection rate of 94% is better than a performance of average hit number 1.97 and detection rate of 93%). Only for the highlighter on the dark desk the detection remains the same. If a single training image yields an equal or better performance than the average from the first two images, we still recommend to use the average because of the risk that the weights from a

Table 3. Experiment 1: Search performance on training sets with weight vectors from 1, 2, and 3 training images obtained with the algorithm in Fig. 2. The performance is presented as the average hit number on the training set and, in parentheses, the percentage of detected targets within the first 10 foci. The best value is highlighted in bold face. Already two training images yield the local optimum in performance and the algorithm stops. The application of these values to a test data set is presented in Tab. 4.

Target	# train im.	av. hit number (and detection rate [%])		
		$\boldsymbol{w}_1$	$\boldsymbol{w}_{(1,2)}$	$\boldsymbol{w}_{(1,..,3)}$
Fire extinguisher	10	1.10 (100%)	**1.00 (100%)**	1.00 (100%)
Key fob	10	1.33 (100%)	**1.00 (100%)**	1.00 (100%)
Name plate	54	1.61 (87%)	**2.04 (94%)**	1.97 (93%)
Highlighter (dark desk)	10	1.50 (100%)	**1.50 (100%)**	1.50 (100%)
Highlighter (bright desk)	10	3.40 (100%)	**2.10 (100%)**	2.40 (100%)

Table 4. Experiment 1: Search performance on test sets with the weight vectors of Tab. 3. The numbers in bold face denote the best performance. Note that the best performance is not always reached at the point that the training proposes: for the highlighter, the best performance is achieved on the dark desk with a single training image and on the bright desk with three training images.

Target	# train im.	# test im.	av. hit number (and detection rate [%])		
			$\boldsymbol{w}_1$	$\boldsymbol{w}_{(1,2)}$	$\boldsymbol{w}_{(1,..,3)}$
Fire extinguisher	10	46	1.14 (100%)	**1.09 (100%)**	1.09 (100%)
Key fob	10	30	1.40 (100%)	**1.23 (100%)**	1.40 (100%)
Name plate	54	238	2.31 (80%)	**2.55 (87%)**	2.28 (86%)
Highlighter (dark desk)	10	30	**1.30 (100%)**	1.37 (100%)	1.37 (100%)
Highlighter (bright desk)	10	30	2.43 (100%)	1.97 (97%)	**2.13 (100%)**

single training do not generalize well enough on the test set. For three images, the performance does not improve any more, on the contrary, the results are worse for some of the examples (name plate, highlighter bright). Therefore, the algorithm stops with the weights vectors $\boldsymbol{w}_{(1,2)}$ as local optima.

In a second step, we apply these weight vectors to test image sets which were disjoint from the training data (Tab. 4). This shows how the system generalizes on unknown data. As expected the performance is in most cases slightly worse than on the training set, since the weights were chosen to fit the training set. However the detection quality is still very high: fire extinguisher, key fob, and the highlighter on the dark desk are detected in all images (detection rate 100%) and the highlighter on the bright desk is missed only in 3% of the images ($\boldsymbol{w}_{(1,2)}$). In the successful cases, the target is detected in average with the 1st or 2nd focus. The most difficult example is the name plate; here, the target is missed in 13% of the images.

It also revealed that the best performance is not always achieved with the same weights as on the training data: For the highlighter on the dark desk, the

best performance is already achieved with the first training image and for the highlighter on the bright desk three training images yield the best performance. This is inevitable since every test set is slightly different and has another combination of weights that fits best for it. Nevertheless, the performance results differ only slightly and the proposed approach yields a good approximation of the optimal performance. Fig. 4 shows some of the test images with some foci of attention.

Experiment 2 In the previous experiment, the background within each image set was similar. Here we show what happens if a target appears on different backgrounds. To achieve this, we combined the image sets of the highlighter on the dark and on the bright desk into one image set. We expected that here more training images are required to yield a local optimum in performance since the training set is inhomogeneous. It turned out that this is usually true but even here the local optimum is sometimes achieved with two images (cf. Tab. 5). We found that it depends on the starting image how many training images are required until the algorithm stops: when the first image was from the bright desk only two images were needed to yield the local optimum. When it was from the dark desk it took longer until the optimum was reached: the best performance was achieved with the average of four training images. The performance was then better than the performance achieved with two images with a bright-desk starting image.

This results from the fact that the search on the dark desk is considerably easier than on the bright one because of the high contrast of the yellow highlighter to the dark desk. Therefore, weights obtained from the bright desk applied to the dark one yield a good performance but not vice versa. If the starting image is from the bright desk, the images that perform badly are also from the bright desk. After taking the average of two training images, the performance does not improve any more. In contrast, if the starting image is from the dark desk, the bad-performing images are from the bright set and $\boldsymbol{w}_{(1,2)}$ is the average of dark and bright. This is repeated and the average of dark and bright yields a performance that excels the former performance after 4 iterations.

In Tab. 6, the computed weights are applied to a test set of 60 images disjoint from the training set. The detection quality is very high: although the target lay on different backgrounds, it is found in all images and in average with the 1st or 2nd focus. Again, it showed that the optimal performance is not always reached for the same weights as the optimal performance on the training set but the training results yield a good approximation of the optimum. Interestingly, with both kinds of weight vectors (bright and dark starting image) the performance on the test set is better than on the training set. Probably this results from a few difficult example images in the training set which decline the average hit number. Note that despite the different results depending on the start image it is not necessary to attach great importance to the choice of this image since the average hit numbers for both cases are very similar. A randomly chosen image will usually suffice.

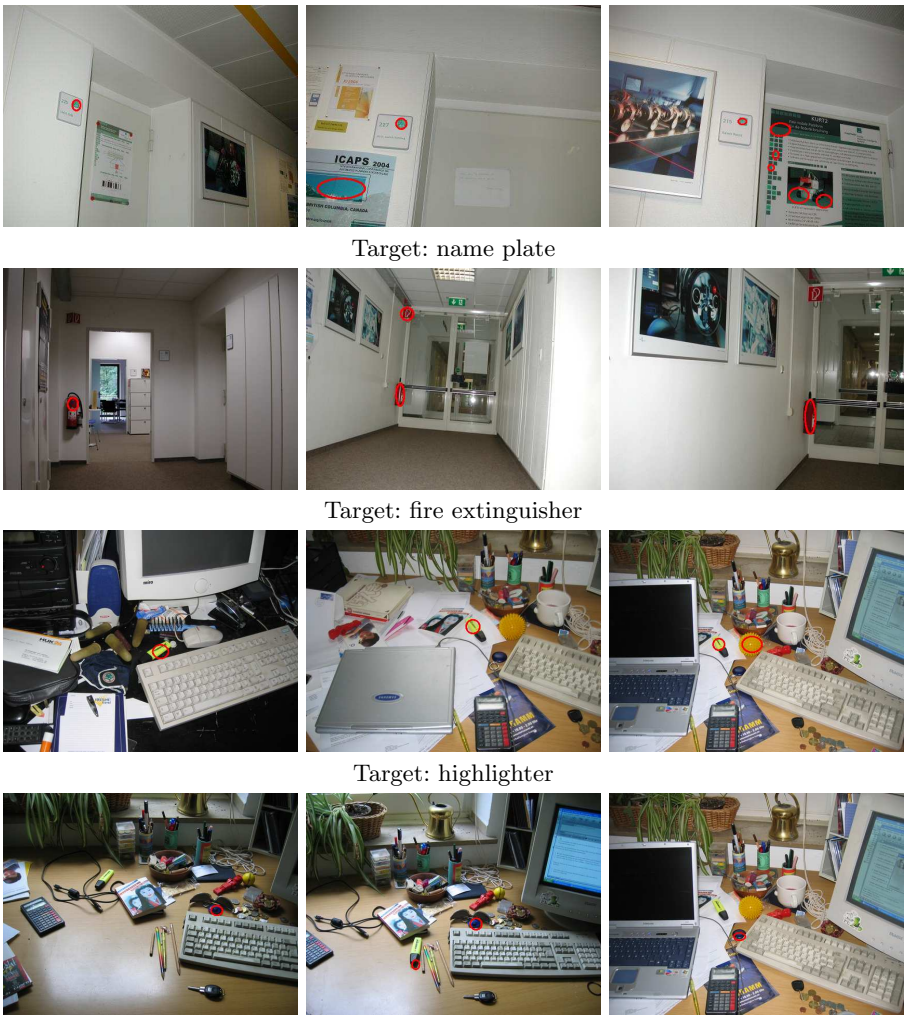


Fig. 4. Some of the results from searching the targets of Fig. 3. The FOAs are depicted by red ellipses. After the target was focused, the search was canceled so the number of depicted foci is equal to the number of required fixations. The hardest example is the one in the upper right corner: the poster shows colors similar to the logo of the name plate and diverts the focus so the target is only detected by the 6th focus. In all other depicted examples the target is found with the first or second focus.

Table 5. Experiment 2: Search performance on a training set (20 images) with different backgrounds: the highlighter (target) lay on a dark and on a bright desk. The weight vectors are obtained with the algorithm in Fig. 2. The performance is presented as the average hit number on the training set and, in parentheses, the percentage of detected targets within the first 10 foci. The best value is highlighted in bold face. The performance depends on the start image: if the start image is from a bright desk (b), the local optimum is reached for 2 training images. If it is from a dark desk (d), 4 images yield the best performance. The application of these values to a test data set is presented in Tab. 6

Target	start	average hit number (and detection rate [%])				
	im.	$w_{i,1}$	$w_{i,(1,2)}$	$w_{i,(1,..,3)}$	$w_{i,(1,..,4)}$	$w_{i,(1,..,5)}$
Highlighter	b	2.45 (100%)	**1.70 (100%)**	1.85 (100%)		
Highlighter	d	2.50 (95%)	1.95 (100%)	1.75 (100%)	**1.55 (100%)**	1.75 (100%)

Table 6. Experiment 2: Search performance on a test set of 60 images with dark and bright backgrounds obtained with the weight vectors of Tab. 5. The numbers in bold face denote the best performance. Note that the best performance is not always reached at the point that the training proposes: for the bright starting image, the best performance is achieved with three training images instead of two.

Target	start	average hit number (and detection rate [%])				
	im.	$w_{i,1}$	$w_{i,(1,2)}$	$w_{i,(1,..,3)}$	$w_{i,(1,..,4)}$	$w_{i,(1,..,5)}$
Highlighter	b	1.80 (100%)	1.53 (99%)	**1.62 (100%)**		
Highlighter	d	1.83 (99%)	1.58 (100%)	1.55 (100%)	**1.48 (100%)**	1.60 (100%)

5 Conclusion

We have introduced a new algorithm that chooses suitable training images for our attention system VOCUS which detects salient regions in images depending on bottom-up and top-down cues. VOCUS provides a method to quickly localizing object candidates to which computationally expensive recognition algorithms may be applied. Thereby, it provides a basis for robust and fast object recognition in computer vision and robotics.

The presented selection algorithm choses the most suitable training images out of an image set. This selection improved the search performance as shown by experiments involving numerous real-world images. We have shown that the chosen image subset is a local optimum and outperforms the use of a single or all training images. In our experiments, less than five training images were sufficient to yield the local optimum. With this approach, VOCUS detects targets robustly and quickly in various scenes: the target was in average among the first three selected regions.

In future work, we plan to utilize the system for robot control. The attention system will determine salient regions in the robot's environment and search for

previously learned objects. Directing the attention to regions of potential interest will be the basic assumption for object detection and manipulation.

Acknowledgements The authors wish to thank Joachim Hertzberg for supporting our work.

References

1. Backer, G., Mertsching, B. and Bollmann, M. Data- and model-driven Gaze Control for an Active-Vision System. *IEEE Trans. on PAMI* **23(12)** (2001) 1415–1429.
2. Corbetta, M. and Shulman, G. L. Control of goal-directed and stimulus-driven attention in the brain. *Nature Reviews* **3** (3, 2002) 201–215.
3. Frintrop, S. VOCUS: A Visual Attention System for Object Detection and Goal-directed Search. PhD thesis University of Bonn Germany (to appear 2005).
4. Frintrop, S., Backer, G. and Rome, E. Goal-directed Search with a Top-down Modulated Computational Attention System. In: Proc. of DAGM 2005 (accepted) Lecture Notes in Computer Science (LNCS) Springer (2005).
5. Hamker, F. Modeling Attention: From computational neuroscience to computer vision. In: Proc. of WAPCV'04 (2004) 59–66.
6. Hunt, R. W. G. *Measuring colour* Ellis Horwood Limited Chichester, West Sussex, England 1991.
7. Itti, L., Koch, C. and Niebur, E. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *IEEE Trans. on PAMI* **20** (11, 1998) 1254–1259.
8. Koch, C. and Ullman, S. Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology* **4** (4, 1985) 219–227.
9. Mitri, S., Frintrop, S., Pervölz, K., Surmann, H. and Nüchter, A. Robust Object Detection at Regions of Interest with an Application in Ball Recognition. In: Proc. of the Int'l Conf. on Robotics and Automation (ICRA '05) (to appear 2005).
10. Navalpakkam, V., Rebesco, J. and Itti, L. Modeling the influence of task on attention. *Vision Research* **45** (2, 2005) 205–231.
11. Neisser, U. *Cognitive Psychology* Appleton-Century-Crofts New York 1967.
12. Palmer, S. E. *Vision Science, Photons to Phenomenology* The MIT Press 1999.
13. Schill, K., Umkehrer, E., Beinlich, S., Krieger, G. and Zetzsche, C. Scene analysis with saccadic eye movements: Top-down and bottom-up modeling. *Journal of Electronic Imaging* **10** (1, 2001) 152–160.
14. Sun, Y. and Fisher, R. Object-based visual attention for computer vision. *Artificial Intelligence* **146** (1, 2003) 77–123.
15. Theeuwes, J. Top-down search strategies cannot override attentional capture. *Psychonomic Bulletin & Review* **11** (2004) 65–70.
16. Treisman, A. M. and Gelade, G. A feature integration theory of attention. *Cognitive Psychology* **12** (1980) 97–136.
17. Tsotsos, J. K. Complexity, Vision, and Attention. In: Vision and Attention, M. Jenkin and L. R. Harris (Eds.) Springer Verlag 2001 chapter 6.
18. Tsotsos, J. K., Culhane, S. M., Wai, W. Y. K., Lai, Y., Davis, N. and Nuflo, F. Modeling Visual Attention via Selective Tuning. *AI* **78** (1-2, 1995) 507–545.
19. Wolfe, J. Guided Search 2.0: A Revised Model of Visual Search. *Psychonomic Bulletin & Review* **1** (2, 1994) 202–238.