Frame Rate Object Extraction from Video Sequences with Self Organizing Networks and Statistical Background Detection

Thiago Craesmeyer Bellardi, Dizan Alejandro Vasquez Govea, Christian Laugier

Frame Rate Object Extraction from Video Sequences with Self Organizing Networks and Statistical Background Detection

Thiago C. Bellardi, Dizan Vasquez and Christian Laugier

***Abstract*— In many computer vision related applications it is necessary to distinguish between the background of an image and the objects that are contained in it. This is a difficult problem because of the double constraint on the available time and the computational cost of robust object extraction algorithms.**

This paper builds upon former work on combining the strong theoretical foundations of clustering with the speed of other approaches. It is based on a novel Self Organizing Network (SON) which has a robust initialization schema and is able to find the number of objects in an image or grid. The main contribution of our extension is that it eliminates the use of a threshold, allowing the algorithm to work on continuous, while having a complexity that remains linear with respect to the number of pixels or cells.

I. INTRODUCTION

Image segmentation is an important and challenging problem in vision. Its goal is to identify homogeneous regions in images as distinct from the background and belonging to different objects. A common approach is to classify pixels on the basis of local features (*e.g.* color, position, texture), and then group them together according to their class in order to identify different objects in the scene.

For the specific problem of finding moving objects from static cameras, the traditional segmentation approach is to separate pixels into two classes: background and foreground. This is called *Background Subtraction* [1] and constitutes an active research domain, the interested reader is referred to [2] for an overview of the field's state of the art.

The output of most background segmentation techniques consists of a bitmap image, where values of 0 and 1 correspond to background and foreground, respectively (*eg* [3], [4], [5]). Having such a bitmap, the next processing step consists of merging foreground pixels to form bigger groups corresponding to candidate objects, this process is known as *object extraction*.

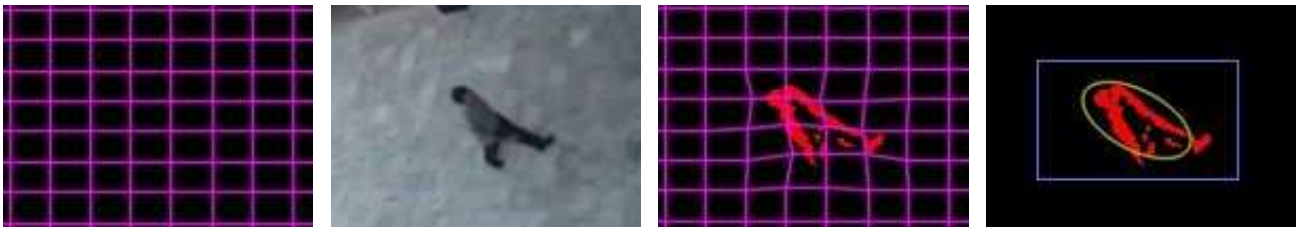
One common procedure to perform object extraction consists of finding 4 or 8-connected components. This is done using efficient algorithms whose time complexity is linear with respect to the number of pixels in the bitmap [6], [7]. A problem with this approach is that it usually produces many small regions which may correspond to noise or to larger regions which failed to merge.

One approach to dealing with this situation is to filter out regions composed of less than a given number of pixels [8]. Although this approach is fast, it has the drawback of assuming that all small regions are noise, which, in many situations, is clearly not the case. A second approach consists of relaxing the neighborhood criterion by assuming, for example, that regions separated by one background pixel are still connected. The usual way of doing this is by pre-processing the bitmap image using morphological operators (*eg* dilation, closing), which have the effect of "thickening" the pixels and "filling in" the holes [9]. Two problems with this approach are the difficulty of finding the appropriate parameters for the operators and the lack of clear physical interpretation of the operators' parameters. A third approach to object extraction is the use of clustering techniques to group pixels. This opens up the possibility of choosing between a plethora [10] of different algorithms having well understood theoretical properties. On the other hand, most of the robust clustering algorithms (*eg* [11], [12]) have three problems when applied to object extraction: a) the number of objects to be found should be known beforehand, b) the algorithms' performance is strongly dependent on the initialization and c) most algorithms are just too complex to be used in systems subject to demanding real-time constraints.

In previous papers, we have proposed a novel clustering approach for object extraction based on Self Organizing Networks (SON). We have applied this algorithm to images [13] and occupancy grids [14], and shown that it is able to produce good results in real time. However, this approach suffered from two drawbacks: a) first, it needed a binary input, which implied the necessity of applying a threshold to the input grid, thus losing information; and b) its complexity depended on the number of input cells and the size of the SON, thus, when the number of input cells was high, the performance of the algorithm degraded. This paper presents an improved version of the algorithm which is able to process continuous input values and, whose complexity only depends on the size of the input grid.

The rest of this paper is structured as follows: in the next section, we present a general approach to object extraction using the *k*-means algorithm and then present the details of

This work has been partially supported by the european BACS Project.
T. C. Bellardi and C. Laugier are with LIG & INRIA Rhone-Alpes, France
thiago.craesmeyer-bellardi@inrialpes.fr
christian.laugier@inrialpes.fr
Dizan Vasquez is with ETH Zurich, Switzerland
vasquez@mavt.ethz.ch



(a) Initial SON (b) Original Image (c) Difference Image and resulting grid (d) Extracted objects

Fig. 1. Approach overview. Enlarged views showing the different steps of our algorithm using CAVIAR data.

the improved algorithm. Sec. III presents an overview of the statistical background/foreground classifier we used. In section IV we explain our implementation of the approach and present some preliminary results against real and synthetic data. Finally our conclusions and some further research directions are presented in §V.

II. Clustering-based object extraction

Assuming that the number of objects k in a bitmap is known, applying clustering to object extraction using the k-means (*i.e.* Expectation-Maximization) [11], [12] algorithm is relatively straightforward:

1) Initialize k cluster centers μ_i with arbitrary values.
2) Assign each foreground pixel to its closest cluster center.
3) Re-estimate every cluster center μ_i as the mean of the points allocated to that cluster.
4) Repeat steps 2-4 until some convergence criterion is met (*e.g.* minimal cluster reassignment).

However, in most cases, the value of k is unknown. Furthermore, even knowing k, the quality of the obtained clustering depends heavily on initialization, since the algorithm trends to get stuck in local minima. Finally every iteration has a cost of $O(N_f k)$ (where N_f is the number of foreground pixels) and, sometimes, many iterations are needed before converging.

In order to deal with those problems, this paper proposes an object extraction approach which combines a Self-organizing Network inspired by the Growing Neural Gas [15] combined with a graph theoretic algorithm used to cut edges in the network's graph.

A. SON-based object extraction

The network is built from $M = W \times H$ nodes connected with undirected edges, arranged in a grid with H rows and W columns (fig. 1(a)). This means that, with the exception of nodes located in the borders, every node i will be connected to four other nodes or neighbors ($neigh(i)$), individually denoted by $u(i)$, $d(i)$, $r(i)$ and $l(i)$ for up, down, right and left, respectively. Every node i has two associated variables: its mean value $\mu_i = (x_i, y_i)$ and an accumulator c_i. In a similar manner, for every edge connecting nodes i and j there will be an accumulator $e_{i,j}$. Besides W and H, the algorithm has two other parameters: $0 < \varepsilon_n < \varepsilon_w \leq 1$ which are the learning rates for node mean adaption.

The following subsections describe the steps that our algorithm performs *for every video frame*, using the bitmap image produced by foreground/background classification as an input.

1) Initialization: The network is initialized by assigning values to all the μ_i node centers in order to form a regular grid (fig. 1(a)). Also, the values of all the weights are set to zero:

$$\{c_i \leftarrow 0, e_{i,j} \leftarrow 0 \,\forall\, i, j \mid i \in [1, M], j \in neigh(i)\} \qquad (1)$$

2) Learning: The learning stage takes as input a bitmap I, typically representing the difference between the last image comming from the camera and a background model (fig.1(b)). Pixels from the input image, are processed starting from the upper-left corner and then sweeping every row from left to right. For every pixel i, its coordinates p_i and value $I(p_i)$ are used to update the SON in four steps:

a. Find the node whose mean value is closest to p_i, referenced as winner (w_i). The search is restricted to a subset of nodes surrounding the previous winner w_{i-1} (*eg* the one corresponding to the previous processed pixel). This subset, that we call the search boundary, is represented by $sbound(i-1)$ (see fig. 2).

$$w_i = \underset{j \in sbound(i-1)}{\operatorname{argmin}} \; \|p - \mu_j - 1\| \qquad (2)$$

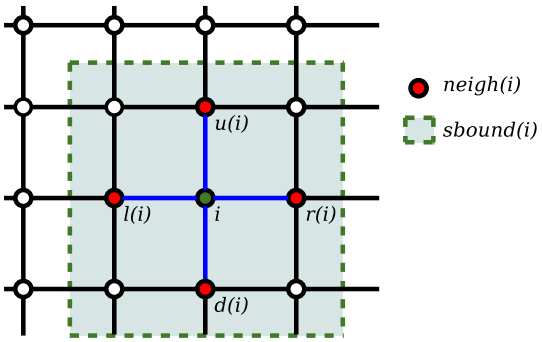


Fig. 2. Neighborhood and search boundary relative to node i

b. Look in $neigh(w_i)$ for the second nearest node to p_i:

$$s_i = \underset{j \in neigh(w_i)}{\operatorname{argmin}} \|p_i - \mu_j\| \tag{3}$$

c. Increment the accumulators e_{w_i,s_i} and c_{w_i} by the pixel value $I(p_i)$:

$$e_{w_i,s_i} \leftarrow e_{w_i,s_i} + I(p_i) \tag{4}$$

and

$$c_{w_i} \leftarrow c_{w_i} + I(p_i) \tag{5}$$

d. Adapt the mean of w_i and all his neighbors:

$$\mu_{w_i} \leftarrow \mu_{w_i} + \varepsilon_w \frac{I(p_i)}{c_{w_i}}(p_i - \mu_{w_i}) \tag{6}$$

$$\mu_j \leftarrow \mu_j + \varepsilon_n \frac{I(p_i)}{c_j}(p_i - \mu_j) \quad \forall j \in neigh(w_i) \tag{7}$$

3) Relabeling nodes: As a result of the learning step, the network adapts its form to represent the objects in the bitmap (fig. 1(c)). The last step of our algorithm, identifies individual objects by assigning a discrete value to every node in the SON, so that nodes having the same label belong to the same node.

Our algorithm finds groups of nodes by merging nodes according to the weight of their common edges $e_{i,j}$. The idea is that a higher value of $e_{i,j}$ corresponds to a higher likelihood that nodes i and j belong to the same object. Under this assumption, it is possible to compute a maximum likelihood estimation of the probability, denoted by $P_{i,j}$, that two nodes "belong together" by using the Laplace law of succession[1]:

$$P_{i,j} = \frac{e_{i,j} + 1}{V + (W-1)H + (H-1)W} \tag{8}$$

where

$$V = \sum_{p \in I} I(p) \tag{9}$$

Also by using the Laplace law of succession, we calculate the value of the uniform link probability distribution, which may be seen as the maximum entropy estimate of $P_{i,j}$ *prior to learning*.

$$\mathbb{U}_{links} = \frac{1}{(W-1)H + (H-1)W} \tag{10}$$

In a similar fashion, the weight c_i is an indicator of the likelihood that node i belongs to an object, it permits us to evaluate how likely it is that the node belongs to the background instead of an object. This is formulated as a probability P_i.

$$P_i = \frac{c_i + 1}{V + WH} \tag{11}$$

[1] $P_{i,j}$ is a notational shortcut introduced for the sake of readability, being rigorous it should be written as $P([O_i = m] \mid [O_j = m])$, where all the O_i variables are binary and $O_i = m$ indicates that node i has been assigned to cluster m. A similar shortcut has been used with P_i for the same reasons.

With the corresponding uniform being:

$$\mathbb{U}_{nodes} = \frac{1}{WH} \tag{12}$$

We use a conventional scanning algorithm to relabel the nodes. The only particularity of our approach is that we use $P_{i,j}$ as the region-merging criterion instead of using colors or other features. Here, we will outline the labeling algorithm, however, the presentation of the complete implementation details is beyond the scope of this paper. The reader is referred to [6], [7] for efficient linear-time ways to implement the algorithm.

The algorithm starts from the upper-left node and proceeds by scanning from left to right and from top to bottom, for every node i the following steps are applied:

a. Assign the label ∞ to i.
b. If $P_{i,l(i)} > \mathbb{U}_{links}$, assign to i the label of $l(i)$ (merge with left region).
c. If $P_{i,u(i)} > \mathbb{U}_{links}$, assign to i the minimum between its current label and the label of $u(i)$. Let a be that minimal label and let b be the label of $u(i)$. Relabel all nodes on the previous rows having label b to a (merge with upper region).
d. If i's label is ∞ assign the next unused label to i (create a new region).

At this point, it is important to highlight the fact that the labels obtained above are just identifiers used to distinguish one region (*ie* object) from the other and that new labels are obtained by incrementing a counter.

4) Computing cluster representations: Having labeled the nodes, a the probability that a pixel, given by its image coordinates p belongs to a cluster m may be represented using a gaussian distribution[2]:

$$P^*(p \mid m) = \mathcal{N}(p; \mu_m^*, S_m^*) \tag{13}$$

The cluster's prior may be used to filter out clusters whose prior is below a given threshold, it is computed as:

$$P_m^* = \sum_{i \in m} P_i \tag{14}$$

Its mean value,

$$\mu_m^* = \frac{1}{P_m^*} \sum_{i \in m} P_i \mu_i \tag{15}$$

And its covariance,

$$S_m^* = \sum_{i \in m} \frac{P_i}{P_m^*} \begin{pmatrix} (x_i - x_m^*)^2 & (x_i - x_m^*)(y_i - y_m^*) \\ (x_i - x_m^*)(y_i - y_m^*) & (y_i - y_m^*)^2 \end{pmatrix} \tag{16}$$

Alternatively, a cluster may also be viewed as a mixture of gaussians, corresponding to individual nodes in the cluster:

$$P^*(p \mid m) = \sum_{i \in m} P_i \mathcal{N}(p; \mu_i, S_i) \tag{17}$$

[2] Hereafter, cluster parameters will be denoted by a superscript asterisk, in order to distinguish them from node parameters

In order to compute the covariance matrices S_i, we use the points located halfway between i and its neighbors (fig. 3):

$$S_i = \sum_{j \in neigh(i)} \frac{P_j}{K} \begin{pmatrix} \left(\frac{x_j+x_i}{2}\right)^2 & \frac{(x_j+x_i)(y_j+y_j)}{4} \\ \frac{(x_j+x_i)(y_j+y_i)}{4} & \left(\frac{y_j+y_i}{2}\right)^2 \end{pmatrix} \quad (18)$$

Where $K = \sum_{j \in neigh(i)} P_j$ is a normalization constant.

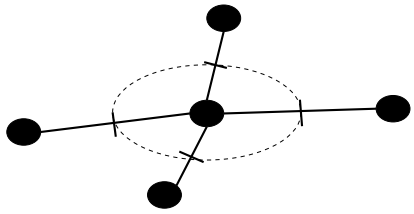


Fig. 3. Estimating the covariance, represented by the ellipse, from the midpoints between a node (center) and its neighbors.

In cases where the algorithm is required to produce interest regions it is often convenient to produce bounding boxes which are slightly larger than the contained object. We have computed the size of these regions using the difference between the maximum and the minimum mean values of the cluster nodes as they were *before learning*[3] this may be regarded as finding the area bounded by nodes which have not been adapted.

B. Complexity Analysis

The changes in the search algorithm, denoted by the equations 2 and 3, represent a complexity change from $O(N_f M)$ to $O(N)$, where N corresponds to the total number of pixels in the image. Noting that now it takes into account all the pixels in the image, instead of just the ones marked as being part of the foreground, and its complexity is independent of the SON grid size. This allows us to weight the pixels background/foreground contribution in a continuous fashion, instead of using a hard threshold, which makes the approach much more robust. The key idea for complexity independence on the SON size is to exploit the fact that the network is processed in a top-bottom, left-right sequence, and to limit the set of nodes in the SON which need to be compared with each pixel by looking in the neighborhood of the last processed pixel.

Thanks to the existence of efficient algorithms, the cost of labeling is linear with respect to the number of nodes in the SON, moreover, the computation of the cluster representation (*i.e.* gaussian parameters, mixture of gaussian parameters and bounding boxes) may be performed at the same time as labeling. Thus, the algorithm's overall complexity is $O(N)$.

III. Statistical background/foreground classifier

The goal of the statistical background classifier is to learn a model of the scene background from a sequence of images. It represents each pixel as an independent normal distribution. For every pixel, the classifier outputs a dissimilarity measure, indicating how different the pixel is from the modeled background, and consequently, how likely it is that pixel belongs to the foreground.

Let us have a given input pixel whose coordinates in the image frame are given by $p = [x,y]$ and whose intensity value is given by $I(p)$. The corresponding background normal will have a mean and standard deviation denoted as μ_p and σ_p respectively. The dissimilarity measure $D(p = fg)$ is computed using the Mahalanobis distance between $I(p)$ and $N(\mu_p, \sigma_p^2)$:

$$D(p = fg) = \frac{|\mu_p - I(p)|}{\sigma_p^2} \quad (19)$$

Once the measure $D(p = fg)$ is computed, the background is updated as follows:

$$\widehat{\mu}_p = D(p = fg)\mu_p + (1 - D(p = fg))I(p) \quad (20)$$

$$\widehat{\sigma}_p = D(p = fg)\sigma_p + (1 - L(p = fg))|I(p) - \mu_p| \quad (21)$$

$$\mu_p \leftarrow \alpha\mu_p + (1-\alpha)\widehat{\mu}_p \quad (22)$$

$$\sigma_p \leftarrow \alpha\sigma_p + (1-\alpha)\widehat{\sigma}_p \quad (23)$$

where α controls the learning rate.

IV. Experimental Results

The algorithm was tested with three different inputs: (a) A binary bitmap obtained by thresholding the absolute difference between the intensity level of the current and previous video frames; (b) A grayscale bitmap obtained from the statistical background/foreground classifier described above, where the intensity represents the mahalanobis distance between the actual frame and the gaussian estimated background (see III); and (c) a thresholded version of the frame used in (b). The SON size and learning factors are the same for the three cases: 20x20 nodes, $\varepsilon_w = 0.1$ and $\varepsilon_n = 0.01$. The tests were conducted using the CAVIAR test case scenarios [16], which consist of a number of video sequences of people moving in the INRIA Lab's entry hall. The videos come with data files containing the ground truth of the sequences, which has been obtained by hand-labeling the images. A typical image of our detector running on one of these videos is shown in fig. 4.

For each frame processed the result of the extraction is compared with the ground truth and the following parameters are computed:

1) Detection ratio (η_{dt})

$$\eta_{dt} = \frac{\text{number of detections}}{\text{number of labeled objects}} \quad (24)$$

[3]This idea was suggested by David Raulo

(a) Original image (b) Grayscale foreground (c) Detection results

Fig. 4. Object detection using a statistical background/foreground classifier. The grayscale foreground is passed as it to the object detector.

2) Matching ratio (η_{match})

$$\eta_{match} = \frac{\text{detection and ground truth matching area}}{\text{ground truth area}} \tag{25}$$

3) False positive ratio (η_{fp})

$$\eta_{fp} = \frac{\text{detected false positive area}}{\text{ground truth area}} \tag{26}$$

4) False negative ratio (η_{fn})

$$\eta_{fn} = \frac{\text{detected false negative area}}{\text{ground truth area}} \tag{27}$$

The mean values for these parameters, obtained from 1042 effective frames processed from the CAVIAR [16] 'Browse2' dataset is shown in table I.

TABLE I

MEAN RESULTS FOR 1042 EFFECTIVE FRAMES PROCESSED FROM THE CAVIAR "BROWSE2" DATASET

input	$\bar{\eta}_{dt}$	$\bar{\eta}_{match}$	$\bar{\eta}_{fp}$	$\bar{\eta}_{fn}$
ideal	1	1	0	0
a	1.45	0.61	7.38	0.38
b	0.96	0.73	7.08	0.27
c	1.95	0.79	9.08	0.20

The obtained result shows that the detector can be combined with more sophisticated foreground classification methods to improve the quality of detection. Is clear that the better the background/foreground classification is, the better will be the result from the clustering, but what is interesting in the comparison between the results from the inputs (b) and (c) is that the method used to extract object performs better with the non-thresholded input.

In respect to computational costs, the object extraction from a 384×288 grayscale image, using a 20×20 SON consumes less than $50msec$ (mean obtained from 1044 frames processed), running on a machine equipped with an Intel Pentium IV processor. To illustrate the algorithm complexity property, we repeated the procedure above using a 100×100 SON and the increase in the processing time kept around $1msec$.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have discussed object extraction from continuous valued bitmaps emphasizing the advantages, but also the three big problems of cluster based algorithms (*ie* need to know the number of objects to be detected beforehand, sensibility to initialization and complexity) and extended previous work on a Self Organizing Network based on the Growing Neural Gas algorithm which solves the above mentioned problems and keeps the strong theoretical properties of clustering algorithms. Our extension permits makes the complexity of the algorithm independent of the size of the underlying SON, and eliminates the need of obtaining a binary image through a threshold.

We have explained the details of our algorithm, and shown how it may be used to find clusters and represent them using gaussians, mixtures of gaussians or bounding boxes.

Finally, we have discussed the experimental results we have obtained by comparing our approach to a ground truth consisting of hand-labeled data. Our results seem to confirm that our approach is fast, robust and general. We have actually applied the approach to find regions of interest that are fed to a pedestrian detection framework using fixed cameras, and obtained very satisfactory results.

Future work includes continuing our experimental work, in particular by improving the background/foreground classifier. Other possibilities include taking into account temporal information by updating the state of the SON when a new input image is available, instead of working in a frame per frame fashion. Finally, we would like to explore the use use of our SON to perform data fusion on a multicamera system installed in a parking lot.

REFERENCES

[1] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell, "Towards robust automatic traffic scene analsis in real-time," in *Proceedings of the Int.Conf. on Pattern Recognition*, Israel, November 1994.

[2] M. Piccardi, "Background subtraction techniques: a review," in *Proceedings of the IEEE Int. Conf. on Systems, Man and Cybernetics*, The Hague, NL, October 2004, pp. 3099–3103.

[3] N. Friedman and S. Rusell, "Image segmentation in video sequences: A probabilistic approach," in *Proceedings of the 13th Conf. on Uncertainty in Artificial Intelligence*, Providence, USA, August 1997.

[4] C. Stauffer and E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747–757, August 2000.

[5] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1337–1442, 2003.

[6] K. Suzuki, I. Horiba, and N. Sugie, "Fast connected-component labeling based on sequential local operations in the course of forward-raster scan followed by backward-raster scan," in *Proceedings of the 15th Int. Conf. on Pattern Recognition*, vol. 2, Barcelona, September 2000, pp. 434–437.

[7] F. Chang, C.-J. Chen, and C.-J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," *Computer Vision and Image Understanding*, vol. 93, no. 2, pp. 206–220, 2004.

[8] L.-H. Chen and J.-R. Chen, "Object segmentation for video coding," in *Proc. of the 15th Int. Conf. on Pattern Recognition*, vol. 3, Barcelona, Spain, September 2000, pp. 383–386.

[9] F. Meyer and S. Beucher, "Morphological segmentation," *Journal of Visual Communication and Image Representation*, vol. 1, no. 1, pp. 21–46, 1990.

[10] A. Jain, M. Murty, and P. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, pp. 265–322, September 1999.

[11] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, L. L. Cam and J. Neyman, Eds., vol. 1. University of California Press, 1967, pp. 281–297.

[12] N. Dempster, A.and Laird, , and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 9, no. 1, pp. 1–38, 1977, series B.

[13] D. Vasquez and T. Fraichard, "A novel self organizing network to perform fast moving object extraction from video streams," in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing (CN), October 2006. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2006/VF06

[14] D. Vasquez, F. Romanelli, T. Fraichard, and C. Laugier, "Fast object extraction from bayesian occupancy grids using self organizing networks," in *Proc. of the Int. Conf. on Control, Automation, Robotics and Vision (ICARCV)*, Singapore (SG), December 2006. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2006/VRFL06

[15] B. Fritzke, "A growing neural gas network learns topologies," *Advances in Neural Information Processing Systems*, 1995.

[16] "Datasets and videos of the european project caviar," http://homepages.inf.ed.ac.uk/rbf/CAVIAR/, July 2003.