Fast object extraction from bayesian occupancy grids using self organizing networks.

Dizan Vasquez, Fabrizio Romanelli, Thierry Fraichard and Christian Laugier
Lab. GRAVIR/IMAG-CNRS
INRIA Rhone-Alpes, France
Email: {vasquezg, fabrizio.romanelli, thierry.fraichard, christian.laugier}@inrialpes.fr

***Abstract*— Despite their popularity, occupancy grids cannot be directly applied to problems where the identity of the objects populating an environment needs to be taken into account (eg object tracking, scene interpretation, etc), in this cases it is necessary to postprocess the grid in order to extract object information.**

This paper approaches the problem by proposing a novel algorithm inspired on image segmentation techniques. The proposed approach works without prior knowledge about the number of objects to be detected and, at the same time, is very fast. This is possible thanks to the use of a novel Self Organizing Network (SON) coupled with a dynamic threshold. Our experimental results on both real and simulated data show that our approach is robust and able to operate at normal camera framerate.

I. Introduction

Occupancy grids [1], [2] are a tool of widespread use in robotics. They decompose the space in a regular cells, and assign to every cell a value, which represents the probability of that cell being ”occupied” by an object. The exact nature of the decomposed space depends on the particular application, it may be, for example, euclidean space or a higher dimension state-space which takes into account velocities, orientations, etc.

Since individual objects may be represented by more than one cell of an occupancy grid, it is not possible to keep track of the objects' identity. Hence applications like tracking, plan recognition or long-term motion prediction, which are inherently object-oriented, need to be able to ”extract” individual objects from occupancy grids. This problem is closely related to image segmentation and background extraction [3], [4] in computer vision, where the goal is to identify homogeneous regions in images as distinct from the background and belonging to different objects. Actually, a bidimensional occupancy grid may be regarded as an image where every cell corresponds to a pixel and their value corresponds to a normalized intensity value.

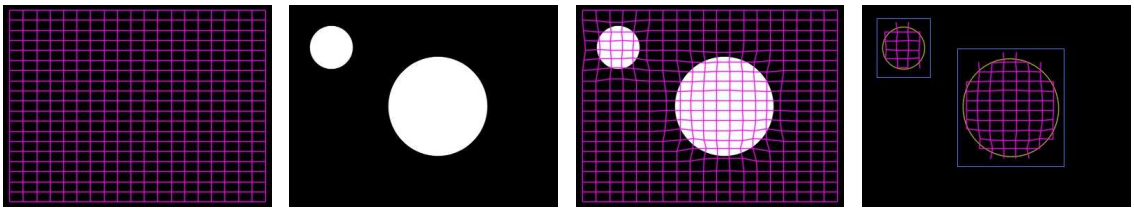
The output of most background segmentation techniques consists of a bitmap image, where values of 0 and 1 correspond to background and foreground, respectively (*eg* [5], [6], [7])[1]. Having such a bitmap, the next processing step consists of merging foreground pixels to form bigger groups corresponding to candidate objects, this process is known as *object extraction*.

[1]It is worth noting, however, that some of these approaches deal internally with more than two classes.

One common procedure to perform object extraction consists in 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 [8], [9]. A problem with this approach is that it usually produces many small regions which may correspond to noise but may also correspond to larger regions which failed to merge.

One approach to dealing with this situation is to filter out regions composed by less than a given number of pixels [10]. 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 preprocessing the bitmap image using morphological operators (*eg* dilation, closing), which have the effect of “thickening” the pixels and “filling in” the holes [11]. Two problems with this approach are the difficulty to find 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 [12] of different algorithms having well understood theoretical properties. In the other hand, most of the robust clustering algorithms (*eg* [13], [14]) 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.

This paper proposes a novel clustering approach for object extraction based on Self Organizing Networks (SON). Although there exist several SON based approaches for segmentation [15], [16], [17], they have been used to perform feature-based pixel classification on images. This is accomplished by performing off-line learning of the classes, and then using the network on-line to classify the pixels of the input image. In contrast, our approach works on occupancy grids. Every time that the occupancy grid is updated, the algorithm is reinitialized and learning is performed – using the position of the grid cells as input – in order to construct a graph with weighted nodes and edges. After learning, a graph-theoretic approach is taken to merge together similar nodes: edges having low



(a) Initial SON configuration (b) Detection bitmap (c) SON after learning (d) Extracted objects

Fig. 1. Approach overview. The images show the different steps of our algorithm using a synthetic occupancy grid.

weights are cut, leaving connected components with high edge values. At the same time, the weights and positions of the nodes may be used to compute a representation of each cluster (*ie* gaussian, mixture of gaussians or bounding box). Our technique specifically addresses the above mentioned problems of clustering based object extraction by proposing a robust initialization scheme, and a mechanism to find the number of objects, all within an $O(N_f M)$ algorithm, where N_f is the number of cells on the occupancy that are above a threshold and M is the number of nodes in the SON. The threshold is computed dynamically in order to minimize the number of cells to be computed.

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 our algorithm. In section III we explain our implementation of the approach and present some preliminary results against pregenerated trajectories. Finally our conclusions and some further research directions are presented in §IV and §V.

II. Clustering-based object extraction

Assuming that the number of objects k in a grid is known, applying clustering to object extraction using the *k*-means (*ie* Expectation-Maximization) [13], [14] algorithm is relatively straightforward:

1) Initialize k cluster centers μ_i with arbitrary values.
2) Assign each grid cell whose probability is above a given threshold to its closest cluster center.
3) Reestimate 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 (*eg* 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 the input probabilities) 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 [18] 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 a counter $c_i \in [0, N_f]$. In a similar manner, for every edge connecting nodes i and j there will be a counter $e_{i,j} \in [0, N_f]$. Besides W and H, the algorithm has other two parameters: $0 < \epsilon_n < \epsilon_w \leq 1$. The meaning of these parameters will be explained in §II-B.

The following subsections (II-A.1 to II-A.5) describe the steps that our algorithm performs *every time that the grid is updated*.

1) Thresholding: The approach used in order to threshold the occupancy grid is dynamic and such that the input set for the SON algorithm will be a sub-set of the whole occupancy grid: the dynamic threshold is computed summing up all the probabilities higher than a value (50% in our experiments) , divided by the number N_t of these probabilities(1).

$$\frac{\sum_{i \in T} p_i}{N_t} \tag{1}$$

where:

$$T = \{i \mid p_i > 50\%\}$$

2) 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(2).

$$\{c_i \leftarrow 0, e_{i,j} \leftarrow 0 \,\forall\, i, j \mid i \in [1, M], j \in neigh(i)\} \tag{2}$$

3) Learning: The learning stage takes the position v of every cell on the occupancy grid whose value if higher than a threshold (1) (fig. 1(b)) and process it in three steps:

a. Determine the two nodes whose means are closest to v:

$$w_1 = \arg \min_{i \in [1,M]} \|v - \mu_i\| \tag{3}$$

and

$$w_2 = \arg \min_{i \in [1,M] \setminus w_1} \|v - \mu_i\| \tag{4}$$

b. Increment the values of e_{w_1,w_2} and c_{w_1}:

$$e_{w_1,w_2} \leftarrow e_{w_1,w_2} + 1 \tag{5}$$

and

$$c_{w_1} \leftarrow c_{w_1} + p_v \tag{6}$$

where p_v is the probability at position v on the occupancy grid.

c. Adapt the mean of w_1 and all his neighbors:

$$\mu_{w_1} \leftarrow \mu_{w_1} + p_v \frac{\epsilon_w}{c_{w_1}}(v - \mu_{w_1}) \tag{7}$$

$$\mu_i \leftarrow \mu_i + p_v \frac{\epsilon_n}{c_{w_i}}(v - \mu_i) \quad \forall i \in neigh(w_1) \tag{8}$$

4) Relabeling nodes: As a result of the learning step, the network adapts its form to cover the objects in the occupancy grid (fig. 1(c)). The last step of 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:

$$P_{i,j} = \frac{e_{i,j} + 1}{N_f + (W-1)H + (H-1)W} \tag{9}$$

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 (*ie* instead of the background), which may be formulated as a probability P_i[2].

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

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 $P_{i,j}$ is used 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 [8], [9] for efficient linear-time ways to implement the algorithm.

[2] $P_{i,j}$ is an abuse of notation introduced for the sake of clarity, 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.

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).

5) Computing cluster representations: Having labeled the nodes, a cluster m may be represented using the gaussian distribution of a point p[3]:

$$P^*(p \mid m) = \mathcal{N}(p; \mu^*_m, S^*_m) \tag{13}$$

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

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

In order to compute the covariance matrices S_i, we use the points located halfway between i an its neighbors (fig. 2):

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

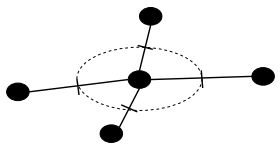


Fig. 2. Estimating the covariance, represented by the ellipse, from the midpoints between a node (center) and its neighbors.

(a) Image from a camera installed on the parking lot with tracker detections

(b) Image from the occupancy grid with the extracted objects

Fig. 3. A typical frame of our system running with real data. The yellow spots are the objects on the grid, the gaussians (ellipses) are estimated by our system.

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

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

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* this may be regarded as finding the area bounded by nodes which have not been adapted.

B. Learning Algorithm Analysis

Having read our learning algorithm, some obvious questions may be: how it differs from the original GNG algorithm? and what are the reasons for these modifications? The following paragraphs will enumerate the differences between GNG and the proposed learning approach and explain the reasons for making these modifications.

1) Addition and deletion of nodes and edges.: One of the main assumptions of the GNG algorithm is that both the topology of the network and the number of nodes (*ie* units) in it are unknown and are going to be determined using a very big number of input samples. In our case, the number of samples per observation is bounded by the number of thresholded occupancy grid input cells. Hence, we have preferred to use a fixed topology and number of nodes M. This should work well on the condition that M is much greater than the maximum expected number of objects in the occupancy grid.

2) Node weight updating: The GNG algorithm is designed to learn from randomly sampled inputs. In our case, the probabilities are processed from top to bottom and from left to right, this results in a skewing phenomenon in which the same nodes get updated many consecutive times and trend to "follow" the direction of sampling. This is due to the fact that the learning factors ϵ_w and ϵ_n are always the same. In order to alleviate this situation we have chosen to use a learning rate which decays with the number of input probabilities c_i that have been assigned to the given node.

3) Edge weight updating: The notion of attaching weights to the links in order to model the probability that two nodes are topologically close is present in the link's age parameter of GNG. However, the way it gets updated is highly discontinuous and, from our point of view, not well suited for modeling it as a probability. Hence, we have profited from the fact that no link deletion/addition takes place in our setting and replaced the age parameter with a counter, which we consider as more appropriate for representing probabilities.

C. Complexity Analysis

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 (*ie* gaussian parameters, mixture of gaussian parameters and bounding boxes) may be performed at the same time than labeling. Thus, the algorithm's complexity is bounded by the learning algorithm complexity which is $O(N_f M)$.

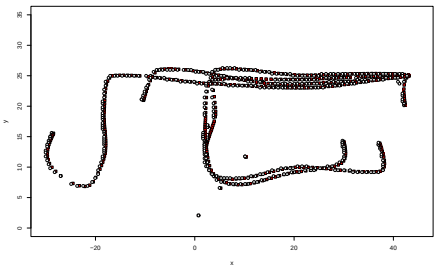
III. Experimental Results

We have tested our approach on the ParkView experimental platform, which consists of a network of cameras used to observe the INRIA laboratory's parking lot. Images coming from the cameras are processed by a visual tracker, the positions of the detected targets are projected into the floor and rasterized to update an occupancy grid[4] which represents the occupancy probabilities for moving objects (*ie* pedestrians and vehicles) moving on the parking lot.

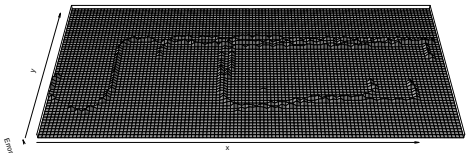
Qualitative results of our experiments seem promising (see fig. 3) but, unfortunately, we are not able at this moment to quantitatively evaluate our experiments due to the lack of ground truth information coming, for example, from a centimetric GPS. Hence, we have designed a series of supplemental experiments using a trajectory simulator which emulates the behavior of objects in the parking lot environment. This allows us to use the ground truth (*ie* original trajectory data) to evaluate the performance of our algorithm.

We have implemented a qualitative test, where the detections over the occupancy grid are plotted together with the

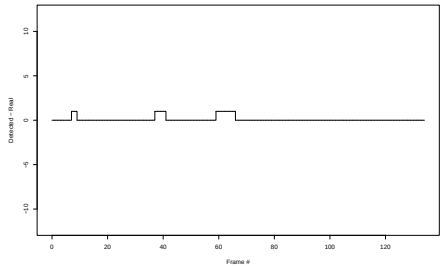
[4]See [19] for further detail on the approach used to update the occupancy grid.



(a) Ground Truth vs. Detections



(b) Measuring detection error



(c) Ground Truth vs. Detections count

Fig. 4. Some results for five objects moving on the scene.

ground truth positions. We have produced several tests, with up to 10 objects moving simultaneously on the scene.

Fig. 4(a) shows the obtained scatter plot for 5 trajectories. In this scene we can see two objects starting their movement from the bottom right, one object starting his trajectory from the right, and the other two objects from the left. As it can be seen by observing the scatter plot, the 5 trajectories are observable, and there are just few detections (when objects are close) which don't coincide with them.

Our second test aimed to measure the detection error, as well as the number times that an object does not get detected. In this experiment we have proceeded in a frame by frame fashion, where for every object in the ground truth, we have searched the closest detection and use the difference to estimate the error.

The obtained results are presented in fig. 4(b), where it may be seen that the detection error is low with respect to the image scale. The graph doesn't show high peaks, meaning that the detection error is always moderate.

In the last experiment we have counted the number of false positives/negatives. In this test we have searched for the number of undetected objects (false negatives) such as for the number of non-existent objects (false positives).

Fig. 4(c) shows that there are just few cases of undetected objects and there is no false positive case at all. This is due to the fact that objects whose trajectories are close together, are detected as a single object, since on the occupancy grid their shapes overlap.

Another example is shown in fig. 5, where we present a more challenging test, where 10 objects are moving simultaneously on the scene, as may be seen from the figures, our algorithm seems to perform well even in this case.

Concerning performance, the average processing time for our algorithm with a 2048 node network on a 128×256 occupancy grid, is $6.7ms$ per frame with an average number of cells above the threshold equal to 170. Although performance may vary depending on the number of input cells.

IV. Conclusions

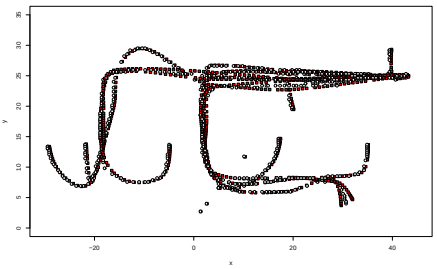
In this paper we have discussed object extraction from occupancy grids, focusing on the advantages, but also on the problems related to cluster based techniques (*ie* need to know the number of the objects to be detected beforehand, sensibility to initialization and complexity) and proposed a novel Self Organizing Network based on the Growing Neural Gas algorithm with the aim of extracting objects from an occupancy grid without knowing *a priori* their number and in a fast and reliable way.

We have explained the details of our algorithm, and shown how it may be used to find clusters in an occupancy grid.

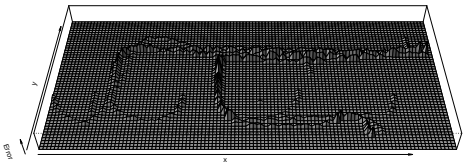
Finally, we have discussed the qualitative experimental results obtained with real data, and given some quantitative measures of the performance of our approach against simulated data. While preliminary, our results seem to confirm that our approach is fast, robust and general.

V. Future Work

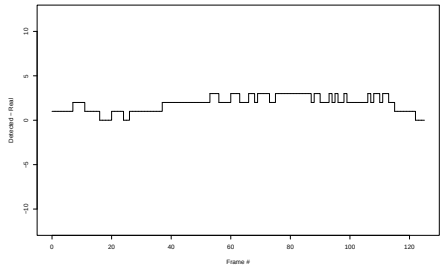
We see are considering several different ways to develop our approach further. In the short term we are planning to



(a) Ground Truth vs. Detections



(b) Measuring detection error



(c) Ground Truth vs. Detections count

Fig. 5. More results for ten objects moving on the scene.

test our approach against other existing techniques for object extraction on occupancy grid.

In the medium term, we will integrate our approach with a multitarget tracking technique (for instance Multiple Hypotheses Tracking), in order to obtain a fully functional tracker over an occupancy grid. We are also planning to explore the application of our approach to perform sensor fusion in the Parkview multicamera platform.

ACKNOWLEDGEMENTS

This work has been partially supported by a Conacyt scholarship. We also want to thank the support of the european BACS and STIC/Asie/FACTS projects as well as David Raulo for his help with the experimental work for this paper and Eric Boniface for his work on the ParkView platform.

REFERENCES

[1] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Magazine*, vol. 9, no. 2, pp. 61–74, July/August 1988, iSSN:0738-4602.

[2] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, pp. 46–57, June 1989, iSSN:0018-9162.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] H. H. Bernd Jahne, Ed., *Computer Vision and Applications*. Academic Press, 2000.

[16] A. Barsi, "Object detection using neural self-organization," in *Proceedings of the XXth ISPRS Congress*, Istanbul, Turkey, July 2004.

[17] Y. Jiang and Z.-H. Shou, "Som ensemble-based image segmentation," *Neural Processing Letters*, vol. 20, no. 3, pp. 171–178, 2004.

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

[19] M. Yguel, O. Aycard, and D. Raulo, "Grid based fusion of offboard cameras," in *IEEE Intelligent Vehicle Symposium*, 2006.