A Novel Self Organizing Network to Perform Fast Moving Object Extraction from Video Streams

Dizan Alejandro Vasquez Govea, Thierry Fraichard

A Novel Self Organizing Network to Perform Fast Moving Object Extraction from Video Streams

Dizan Vasquez and Thierry Fraichard
Lab. GRAVIR/IMAG-CNRS
INRIA Rhone-Alpes, France
Email: {vasquezg,thierry.fraichard}@inrialpes.fr

***Abstract*— Image segmentation is a critical task in computer vision. In the context of motion detection, a very popular segmentation approach is background substraction which consists in classifiying the pixels as background and foreground. Then, the foreground pixels are grouped together to find objects, this task is known as object extraction. There are several different approaches to object extraction (eg connected component labeling, morphological operators, size thresholding and clustering) amongst them, cluster based approaches are, probably, the ones with a stronger theoretical foundation. However, their application to object extraction is difficult because of three problems: a) need to know the number of objects to be detected beforehand, b) high sensibility to initialization due to a trend to get stuck in local minima and c) high complexity which difficults their application in real-time.**

This paper proposes an algorithm which aims to combine the strong theoretical foundations of clustering with the speed of other approaches. This is possible due to the introduction of a novel Self Organizing Network (SON) which has a robust initialization schema and is able to find the number of clusters in the image. The algorithm has a time complexity of order NM where N is the number of foreground pixels in the image and M is the number of nodes in the SON.

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 (*eg* color, position, texture), and then grouping 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 Substraction* [1] and constitutes an active research domain, the interested reader is referred to [2] for an interesting overview of the field's state of the art.

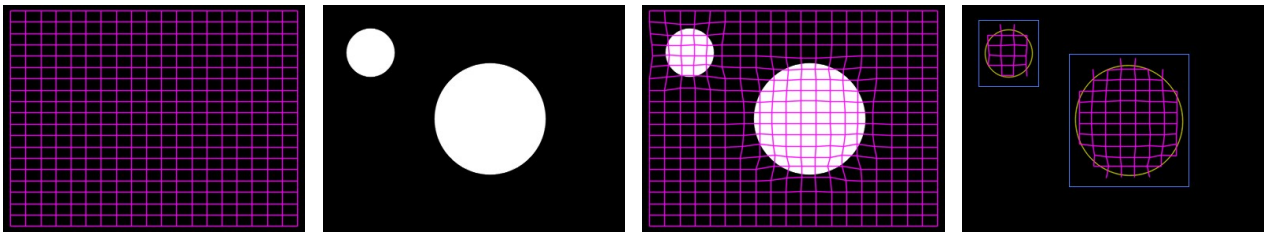
The output of most background segmentation techniques consists of a bitmap image, where values of 0 and 1 correspond to background and foreground, respectively (*eg* [3], [4], [5]). Having such a bitmap, the next processing step consists of merging foreground pixels to form bigger groups corresponding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

This work has been partially supported by a Conacyt scholarship. We also want to thank the support of the CNRS Robea ParkNav and the Lafmi NavDyn Projects.

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 [6], [7]. 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 [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 preprocessing 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 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 [10] of different algorithms having well understood theoretical properties. In 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.

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 [13], [14], [15], they have been used to perform feature-based pixel classification. 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 the bitmap produced by the foreground/background segmentation layer. For every frame, the algorithm is reinitialized and learning is performed – using the position of the foreground pixels 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 weights are cut, leaving



(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 bitmap.

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 foreground pixels and M is the number of nodes in the SON.

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 hand-labeled data. Finally our conclusions and some further research directions are presented in §IV.

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 (*ie* 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) 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 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 [16] 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.4) 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(1).

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

2) Learning: The learning stage takes every foreground pixel p of the input bitmap (fig. 1(b)) and process it in three steps:

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

$$w_1 = \arg \min_{i \in [1,M]} \|p - \mu_i\| \tag{2}$$

and

$$w_2 = \arg \min_{i \in [1,M] \setminus w_1} \|p - \mu_i\| \tag{3}$$

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

$$e_{w_1,w_2} \leftarrow e_{w_1,w_2} + 1 \tag{4}$$

and

$$c_{w_1} \leftarrow c_{w_1} + 1 \tag{5}$$

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

$$\mu_{w_1} \leftarrow \mu_{w_1} + \frac{\epsilon_w}{c_{w_1}}(p - \mu_{w_1}) \tag{6}$$

$$\mu_i \leftarrow \mu_i + \frac{\epsilon_n}{c_i}(p - \mu_i) \quad \forall i \in neigh(w_1) \tag{7}$$

3) Relabeling nodes: As a result of the learning step, the network adapts its form to cover the objects in the bitmap (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[1]:

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

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*.

$$\mathbb{U}_{links} = \frac{1}{(W-1)H + (H-1)W} \tag{9}$$

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.

$$P_i = \frac{c_i + 1}{N_f + WH} \tag{10}$$

With the corresponding uniform being:

$$\mathbb{U}_{nodes} = \frac{1}{WH} \tag{11}$$

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 [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).

[1] $P_{i,j}$ is a notational shortcut introduced, 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.

4) Computing cluster representations: Having labeled the nodes, a cluster m may be represented using the gaussian distribution of a point p[2]:

$$P^*(p \mid m) = \mathcal{N}(p; \mu^*_m, S^*_m) \tag{12}$$

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

$$P^*_m = \sum_{i \in m} P_i \tag{13}$$

Its mean value,

$$\mu^*_m = \frac{1}{P^*_m} \sum_{i \in m} P_i \mu_i \tag{14}$$

And its covariance,

$$S^*_m = \sum_{i \in m} \frac{P_i}{P^*_m} \begin{pmatrix} (x_i - x^*_m)^2 & (x_i - x^*_m)(y_i - y^*_m) \\ (x_i - x^*_m)(y_i - y^*_m) & (y_i - y^*_m)^2 \end{pmatrix} \tag{15}$$

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

$$P^*(p \mid m) = \sum_{i \in m} P_i \mathcal{N}(p; \mu_i, S_i) \tag{16}$$

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

$$S_i = \sum_{j \in neigh(i)} \frac{P_j}{K} \begin{pmatrix} \left(\frac{x_j + x_i}{2}\right)^2 & \frac{(x_j + x_i)(y_j + y_j)}{4} \\ \frac{(x_j + x_i)(y_j + y_i)}{4} & \left(\frac{y_j + y_i}{2}\right)^2 \end{pmatrix} \tag{17}$$

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*[3] this may be regarded

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

[3] This idea was suggested by David Raulo

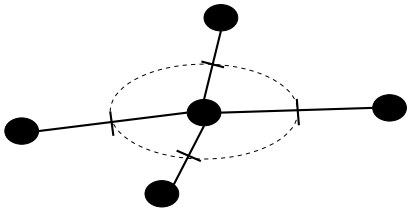
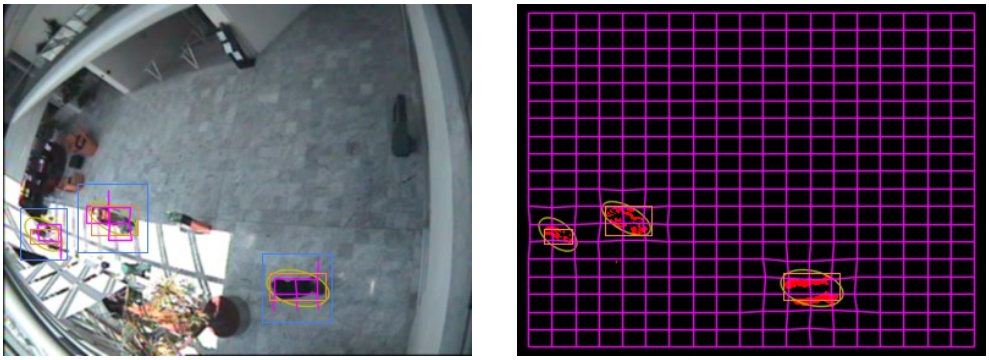


Fig. 2. Estimating the covariance, represented by the ellipse, from the midpoints between a node (center) and its neighbors.



(a) Hall image (b) Classification bitmap

Fig. 3. A typical frame of our system running with a CAVIAR video. The small boxes correspond to ground truth, the bigger ones and the gaussians (ellipses) are estimated by our system.

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 frame is bounded by the number of pixels in the input bitmap. 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 image.

2) Node weight updating: The GNG algorithm is designed to learn from randomly sampled inputs. In our case, the pixels 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 pixels 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 form modelling 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

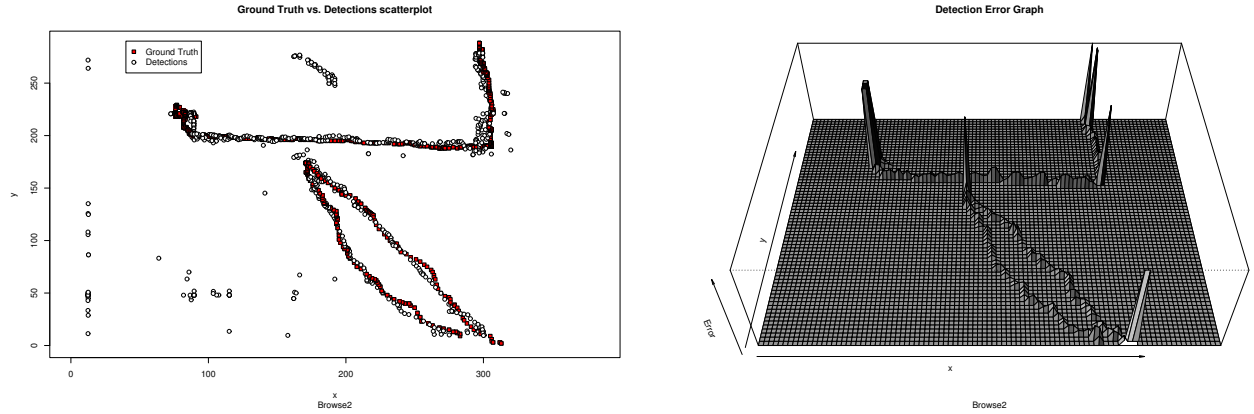
Thanks to the existence of efficient algorithms, the cost of labeling is linear with respect to the number of nodes in the SON, moreover, the computation of the cluster representation (*ie* gaussian parameters, mixture of gaussian parameters and bounding boxes) may be performed at the same time than labeling. Thus, the algorithm's complexity is bounded by the learning algorithm complexity which is $O(N_f M)$.

III. Experimental Results

Our approach has been implemented using a difference bitmap as input, the bitmap has been obtained by thresholding the absolute difference between the intensity level of the current and previous video frames. In order to reduce noise, we have preprocessed every frame by applying a gaussian blur filter. We have fixed the learning factors to $\epsilon_w = 0.1$ and $\epsilon_n = 0.01$. With such settings, some preliminary tests have been conducted using the CAVIAR test case scenarios [17], 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. 3.

First, we have implemented a qualitative test, by simply plotting the ground truth positions together with the detections obtained with our approach. Fig. 4(a) shows the obtained scatterplot on the Browse2 dataset. In this scene we may observe two people moving: one comes from the lower right corner towards the center and then back. The other one goes from the upper left to the upper right and then exits at the top. By observing the scatterplot, this two trajectories are clearly observable, however, there are many detections which do not coincide with them. By inspecting the video, we have identified two additional detection sources:

1) A shelf containing white documents, located in the upper middle. The high luminance values of the documents, combined with the camera's jitter are a source of coherent noise which gets past the gaussian blur.



(a) Ground Truth vs. Detections

(b) Measuring detection error

Fig. 4. Some preliminary results against human classified data.

2) A number of persons moving in front and behind the desk (lower middle) and in the cafeteria (center left) that were not labeled in the ground truth because they were not walking but that got detected because the upper part of their body was moving.

Our second test aimed to measure the precision of our detection, as well as the number times that an object does not get detected. Due to the fact that, as discussed above, there are unlabeled moving objects in the videos, no tests have been implemented to measure the number of false positives. 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 used the distance to estimate the error. In the cases where no such detection is available (false negatives) a big constant value has been assigned as the detection error.

The results obtained for dataset Browse2 are presented in fig. 4(b). as it may be seen, the detection error is low with respect to the image scale. Moreover, the high peaks in the graph correspond to places where the subjects have stopped walking for a while, which is logical given that we are using a simple last-frame difference to detect motion, meaning that the subject is imediatly "lost" when it stops moving. Maybe the most interesting observation about this graph is that peaks are found only in the stop points, which means that they were always detected when moving.

At the end of this paper, we have included some other runs against CAVIAR data, which we do not have enough space to comment here.

We have also performed some qualitative tests on a set of outdoor videos made public by KOGS-IAKS Universitaet Karlsruhe [18]. Unfortunately, since no ground truth is available, the tests consists only in visual inspection of the results. Anyway, the algorithm seems to perform very well even in the presence of fog (fig. 6).

Concerning performance, the average processing time for our algorithm with a 400 node network on a 384×288 image, is $14msec$. Although performance may vary depending on the number of foreground pixels.

IV. Conclusions and Future Work

In this paper we have discussed object extraction from binary 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 initialisation and complexity) and proposed a novel Self Organizing Network based on the Growing Neural Gas algorithm with the aim of alleviating the above mentioned problems while keeping the strong theoretical properties of clustering algorithms.

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. While preliminary, our results seem to confirm that our approach is fast, robust and general.

Future work includes continuing our experimental work, in particular by testing our approach against other existing techniques. We are also working to integrate our approach as

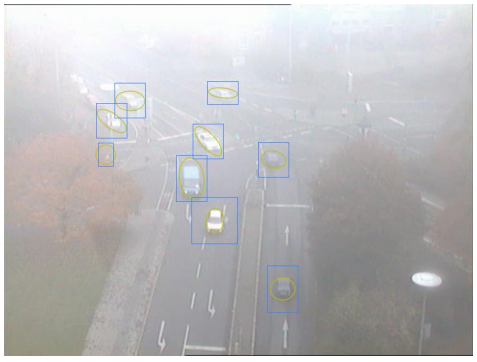
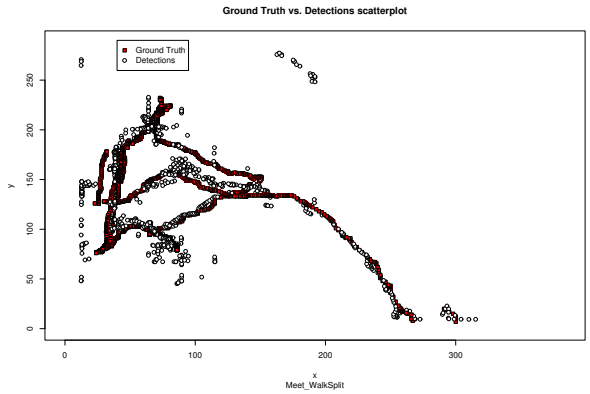
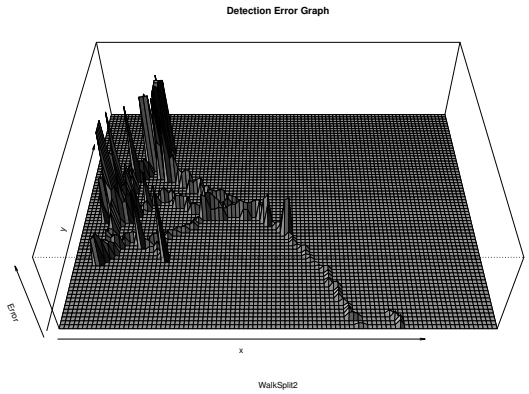


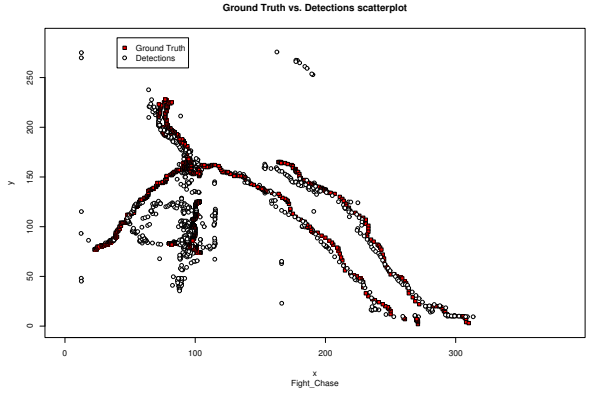
Fig. 6. An example of outdoor detection in the presence of fog.



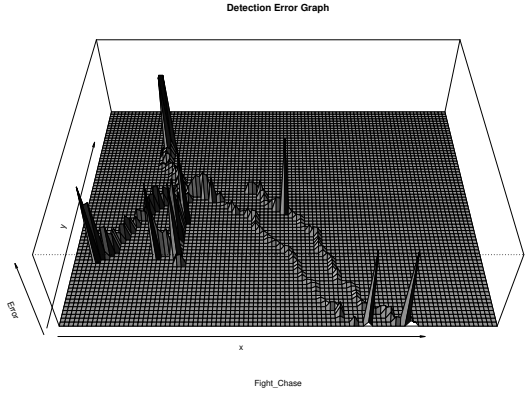
(a) Ground Truth vs. Detections



(b) Measuring detection error



(c) Ground Truth vs. Detections



(d) Measuring detection error

Fig. 5. More results against human classified data.

a region of interest detector for a tracker which uses Boosting techniques for Pedestrian recognition. At this moment, we are finishing up our first prototype of the complete chain.

In the mean term, we are planning to apply our technique to bayesian occupancy grids. Another possible extension is the 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] H. H. Bernd Jahne, Ed., *Computer Vision and Applications*. Academic Press, 2000.

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

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

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

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

[18] "Kogs-iaks universitaet karlsruhe image sequence server," http://i21www.ira.uka.de/image_sequences/.