CROWD ANALYSIS BY USING OPTICAL FLOW AND DENSITY BASED CLUSTERING

Francesco Santoro, Sergio Pedro, Zheng-Hua Tan and Thomas B. Moeslund†

Department of Electronic Systems   †Department of Media Technology
Aalborg University, Njels Jernes Vej 12, 9220 Aalborg, Denmark
francesco.santoro.00@gmail.com; sffpedro@es.aau.dk; zt@es.aau.dk; tbm@imi.dk

ABSTRACT

In this paper, we present a system to detect and track crowds in an image sequence captured by a camera. In the first step, we compute optical flows by means of pyramidal Lucas-Kanade feature tracking. Afterwards, a density based clustering is used to group similar vectors. In the last step, a crowd tracker is applied to each frame, allowing us to detect and track the crowds. The output of the system is given as a graphic overlay, i.e. arrows and circles with different colors are added to the original images to visualize crowds and their movements. Evaluation results show that the system is capable of detecting certain events in the crowds, such as merging, splitting and collision.

1. INTRODUCTION

Security systems are getting more and more important nowadays. With the growth of threats, companies tend to invest more capital in surveillance and security companies. These security companies have a tendency to develop more digital surveillance applications and deploy less human resources, especially with the global economic crisis. Moreover, not only surveillance is important, but also systems capable of recognizing how people act in public scenarios and analyzing the information coming from it. For instance, if there is a moment when the speed of crowd momentarily goes to zero, maybe there are some event happening in that space. The capability of detecting where the group of people is moving is useful for urban design. For instance, the area where many people are passing frequently should not be designed as an enclosed space. For those reasons, crowd analysis has received more and more attention from technical and social research disciplines.

In this paper, we propose a system that detects multiple crowds and their movements. The system detects crowds with different shapes and motions and tracks them over time. Moreover, it has the function to compute statistics from the collected tracking information. The paper is organized as follows. In Section 2 relevant work is discussed. In Section 3 we introduce our system, which consists of the following blocks: Optical Flow Computation, Block Partitioning, Density Based Clustering and Crowd Tracking. We make an evaluation of our system in Section 4 and finally we conclude and discuss possible future work in Section 5.

2. RELATED WORKS

One of the most intuitive techniques for detecting object is background subtraction. It detects the foreground objects as the difference between the current frame and an image of the scene's static background. Several algorithms using this technique are presented in [1] and [2]. Nevertheless, this technique is sensitive to illumination and motion changes. For instance, camera oscillations or high-frequency background objects may disturb the results of this technique. Other methods combine statistics with knowledge on the moving object [3]. Using real-time segmentation of moving regions, an improvement is presented in [4]. According to the solutions proposed, background subtraction gets accurate, but they are highly computational demanding.

Block matching technique [5] divides images into small blocks. For each reference block, a search is made over all shifted versions of that block within a rectangular region in the next frame, known as the search window. The candidate block with the minimum distortion from the reference block gives the estimated motion of the reference block. Many criteria exist for the distortion measurement between reference and candidate blocks. In [6] the mean absolute error is used, which offers a good trade-off between complexity and efficiency.

Optical flow methods, that use differential techniques, are based on the hypothesis that brightness of a particular moving point is constant over time. In order to calculate optical flows a lot of algorithms are proposed such as Horn and Schunck's Method [7] and Lucas and Kanade's Method [8]. These methods suffer from problems of accuracy and illumination changes. In order to solve these problems some improvements are presented. Among them, the most important one is the pyramidal implementation of Lucas-Kanade's Method [9], that allows to detect motions with different speeds.

Andrade [10] characterizes crowd behaviour by observing the optical flow associated with the crowd and uses unsupervised feature extraction to encode normal crowd behaviour. Other techniques have been recently presented. A relevant one [11] is the introduction of self-organizing maps for the visualization of crowd dynamics and to learn models of the dominant motions of crowds in complex scenes. An useful and interesting technique is the detection of major flows and events in crowd scenes, using a Direction Model constructed from Von Mises distributions applied to the orientation of the optical flow vectors [12].

3. A CROWD ANALYSIS SYSTEM

In order to find out which part of the input frames are moving, we use the *KLT feature tracker* [13] where motion detection is computed by using the pyramidal Lucas-Kanade optical flow method, based on the *Shi and Tomasi Corner Detection Algorithm* [14]. The pyramidal implementation allows us to obtain more robustness against big movements. Besides, the method is fast enough to allow us to obtain real-time results.

An important remark is the fact that we are going to deal with crowds that can assume different shapes. For that, a density based clustering is well suited to group the different motion vectors into clusters. An interesting solution is the DBSCAN algorithm [15], that has the capability to work without knowing how many clusters it has to discover and at the same time it can find clusters with different shapes.

Figure 1 shows the basic computation flow of our system. The system takes raw sequential frames as input and gives crowd detection in a graphic overlay as output.

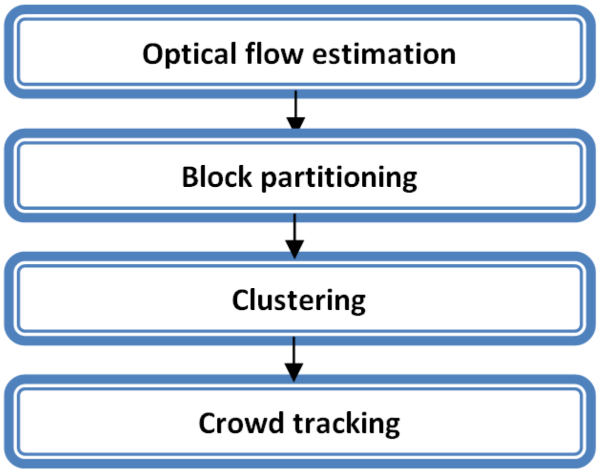

Figure 1: Data flow

3.1 Optical Flow Computation

This step takes two subsequent frames as input F_k and F_{k+1}: first F_k is used to detect strong corners (this allows us to reduce the dimensionality, since we leave out pixels that are not strong corners), and then we exploit F_{k+1} to compute optical flow of these corners. The optical flow estimation returns a matrix A_k :

$$\begin{pmatrix} V_{11} & \cdots & V_{1j} & \cdots & V_{1Q} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ V_{i1} & \cdots & V_{ij} & \cdots & V_{iQ} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ V_{P1} & \cdots & V_{Pj} & \cdots & V_{PQ} \end{pmatrix}$$

where P and Q are the horizontal and the vertical dimension of the input frames, respectively, and $V_{ij} = (X_{ij}, Y_{ij}, M_{ij}, \alpha_{ij})$ is the motion vector related to the ij pixel. For each V_{ij}:

- X_{ij} and Y_{ij} are the X and Y coordinate at frame F_{k+1} of pixel ij of frame F_k;
- M_{ij} is the magnitude of the vector computed as the Euclidean distance between point (i, j) and point (X_{ij}, Y_{ij})
- α_{ij} is the motion direction of pixel ij

Since we compute motion of strong corners only, many V_{ij} vectors will be *null* vectors. In other hand, M_{ij} and α_{ij} could have been not included in V_{ij}, but since they are used several times during the crowd tracking stage, we have decided to use them in order to improve the efficiency of the crowd tracker.

3.2 Block Partitioning

In order to reduce dimensions ($P \times Q$ matrix is too big) and noise (optical flow can be noisy if we consider single vectors), we compute a matrix $\tilde{A}_k$ by applying a window $W \times W$, such that $\tilde{A}_k$ is:

$$\begin{pmatrix} \tilde{V}_{11} & \cdots & \tilde{V}_{1j} & \cdots & \tilde{V}_{1\tilde{Q}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \tilde{V}_{i1} & \cdots & \tilde{V}_{ij} & \cdots & \tilde{V}_{i\tilde{Q}} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \tilde{V}_{\tilde{P}1} & \cdots & \tilde{V}_{\tilde{P}j} & \cdots & \tilde{V}_{\tilde{P}\tilde{Q}} \end{pmatrix}$$

where:

- $\tilde{P} = P/W$
- $\tilde{Q} = Q/W$
- $\tilde{V}_{ij}$ is the vector sum of all vectors in a window, centered at pixel ij

In order to reduce the error made in crowd detection with respect to the benchmark data (see section 4), we choose empirically $W = 11$. We obtain a faster cluster computation (see section 3.3) by raising the value of W but we loose information about motion vectors. Similarly, we reduce the approximation but we need more time to find out clusters by lowering W, because the resulting $\tilde{A}_k$ will be too big.

3.3 Density Based Clustering

Each vector belonging to the $\tilde{A}_k$ matrix represents the motion vector of a certain block of the image plane. These vectors need to be grouped in order to form a crowd, i.e. by applying clustering techniques [16]. A special attention has to be paid to the definition of a cluster in our case. Generally speaking a cluster is a collection of objects/data that have:

- High internal similarity: objects in a given cluster are very similar between themselves;
- Low external similarity: given two objects taken from two different clusters, they have low similarity between themselves.

The neighbourhood function is defined as follows: given two vectors Z_1 and Z_2 they are said to be neighbours if the following conditions are all jointly met:

- Their initial points are no more distant than ε, empirically set to 10.
- Their directions are similar.
- Their magnitudes are similar.

The *similarity* introduced above means that we are going to define some acceptance thresholds for direction and magnitude:

- two vectors V_1 and V_2 have similar directions if the absolute value of the difference between their angles, α_1 and α_2 is not greater than 25 degrees.
- two vectors V_1 and V_2 have similar magnitude if the absolute value of the difference between their magnitudes is not greater than 10.

Moreover, this definition of similarity allows us to recognize splitting and merging crowds and to deal with collapses. The above defined thresholds are strictly related to the mutual position of the camera and to the crowds to be chased. A closer camera will be more sensitive to small movements and to noise as well, so it would be more suitable to raise these thresholds. A farther camera will not notice small differences between vectors, so we will have to reduce acceptance thresholds.

3.4 Crowd Tracking

In our implementation, each crowd has an ID. During the crowd detection process it could happen that the same crowd is given two different IDs across different frames. This is due to the fact that the crowds moving on the image plane are merging and splitting, and it is often difficult to recognize the same crowd after several frames. Besides, our model is stateless, i.e. it cannot accomplish this *ID fixing* task. We solve this problem by:

- Recognizing the same crowd across the whole frame sequence and assign it an unique ID, and
- Removing the crowds that are probably results of optical flow errors or noise.

In order to realize the first step, we use a similarity function. The idea is to label two crowds with the same ID if they are very similar. The formalization of the similarity function is very important. This is a convex combination between: distance between points of application, i.e. distance between center of mass (B); difference between directions (D); and difference between areas (A). Let us remember that direction of a cluster means the average of all pixel's direction in that cluster. Thus, given two crowds, C1 and C2, the similarity function S(C1, C2) will be:

$$S(C_1,C_2) = \alpha_1 B + \alpha_2 D + \alpha_3 A \tag{1}$$

where

$$\sum_{i=1}^{3} \alpha_i = 1 \tag{2}$$

by definition of convex combination and with the result normalized to a range of [0,1]. The α_i parameters have been set as follows:

- $\alpha_1 = 0.7$; $\alpha_2 = 0.2$; and $\alpha_3 = 0.1$

In order to recognize and assign an unique ID for any crowd in different frames, we need to store a buffer with L crowds recognized in the L previous frames. In fact if we have worked only based on the last frame, we could not recognize crowds that disappear for a frame or two because of noise or occlusion.

Since we have defined a similarity function between two crowds, we can recognize crowds across several frames. Given two frames F_1 and F_2 let us define $\Phi_1 = (C_1,C_2,...,C_i,...,C_n)$ and $\Phi_2 = (K_1,K_2,...,K_j,...,K_m)$ the sets of crowds that are contained in F_1 and F_2, i.e. all the crowds detected by means of the above described method. For each crowd C_i, belonging to Φ_1 we want to find in Φ_2 one and only one K_j that best matches, i.e. the crowd K_j that is more similar to C_i than all the others contained in Φ_2.
We define a function $M(C_i)$ which returns the crowd K_j that best matches with C_i; in other words we want the crowd K_j, taken from Φ_2, that is most similar to C_i.
$M(C_i)$ is defined as follows:

$$M(C_i) = argmax_{K_j}(S(C_i,K_j)) \tag{3}$$

and it is computed for each $C_i \in \Phi_1$.
To prevent errors in detecting crowds across different frames, we take into account only those values of $S(C_i,K_j)$ that are greater or equal to a given threshold T. This way, C_i is not matched if there is no K_j such that the similarity between them is at least T. Empirically T has been set to 0.7. Moreover we want to assign to each C_i one and only one K_j: thus, if there are two crowds C_a and C_b, and a crowd K_z such that $M(C_a) = M(C_b) = z$, K_z will be matched to the crowd that is more similar to it. After we have found all the best matching couples (C_i,K_j), we say that:

- if there are any C_i that are not yet matched, they have disappeared across time, or have merged.
- similarly, if there are any K_j that are not yet matched, they are recognized to be new crowds that have entered the scene, or have split from an existing one.

Lastly we need to remove crowds originating from noise. In order to distinguish between a real crowd and a false one simply observe how long it lasts in the frame sequence: a false crowd is supposed to last a short time and thus to be avoided. Since there can be situations where a big crowd is seen as two crowds that move close to each other for a few frames, we do not want to detect a splitting event. Thus, if a crowd lasts for less than 4 frames, it is not considered as a crowd and hence it is removed.

4. EVALUATION

The application has been tested against PETS2009 benchmark data [17]. We focused on three kind of challenges:

- Crowd merging event: two or more groups become close enough to be considered a single group from that point on;
- Crowd splitting event: persons belonging to a single crowd start walking apart so that they cannot be considered a single group anymore;
- Crowd collision event: two or more crowds walking in two different directions merge for some frames and then split up again.

The first two types of events are detected by analyzing the crowds' orientations and distances; on the other hand, for the third category, we have used a tracker module to recognize crowds again after merging and splitting. Then we collect the following information focusing on three variables:

- *Delay*, i.e. the error in detecting an event (splitting or merging), expressed in terms of delay between the frame where the event happens and the one in which our system recognizes it;
- True positive rate (*TPR*), the ratio between the number of events recognized and the number of events that actually happens in the scene;
- False negative (*FN)*, i.e. the number of false events detected by our system.

In the following different scenarios will be presented. Each scenario is structured as follows:

- **Challenge type**: Merging, splitting, crowd chasing after collision;
- **Description**: Short description of the test case;
- **Sequence**: Source sequence name (from PETS2009 benchmark data) and its length in frames.

4.1 Scenario 1

Challenge type: Crowd merging [18].

Description: This sequence contains a densely grouped crowd and a single person joining it.
Sequence: (Pets 2009) S1.L2.14-06.View1 - 201 frames.

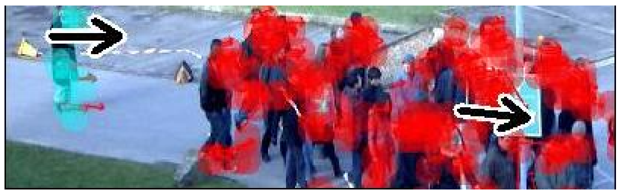

Figure 2: The single person is still detached from the crowd. Each arrow represents the movement of a cluster.

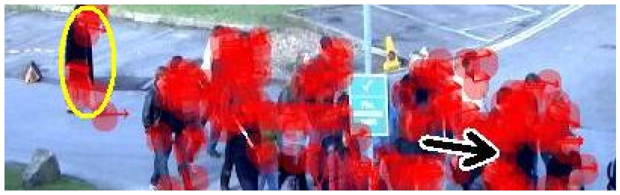

Figure 3: The single person has just merged with the crowd

In this scenario (Figs. 2 and 3), we have the following results: capable of tracking correctly a crowd merging (TPR=1.0), without any delays (Delay=0). Moreover, no false events are detected (FN=0.0).

4.2 Scenario 2

Challenge type: Crowd splitting [18].
Description: A single crowd is moving diagonally. At a point it splits up into 3 different crowds.
Sequence: (Pets 2009) S1.L2.14-31.View1 - 131 frames

In Fig. 4 the first splitting is shown: the left marked part of the crowd is the portion that is currently splitting. In Fig. 5, the second splitting is shown.

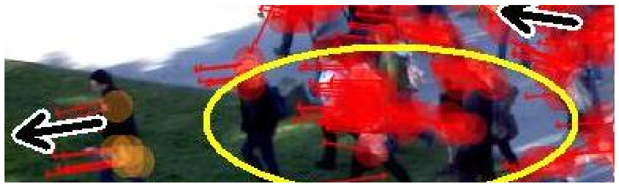

Figure 4: First splitting detection for scenario 2

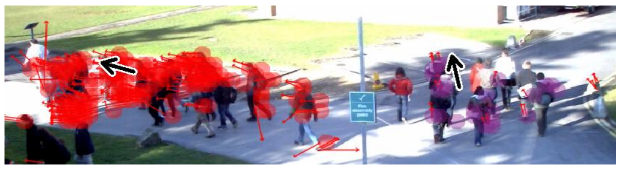

Figure 5: Second splitting detection for scenario 2

In the first splitting, the crowd that is departing from the bigger one is partially overlapped on the bigger one. This happens because we are mapping a 3D space into 2D, causing some delay in the first event. In this scenario, the whole image plan is covered by shadow, i.e it has a low spatial color gradient, causing delays in the detection of both splitting. Nevertheless, we have obtained the following results for this scenario: a good capacity of tracking a crowd splitting (TPR = 1.0) with some delay (Delay = 4) due to the overlapping effect introduced above. No false events were detected (FN = 0.0).

4.3 Scenario 3

Challenge type: Occlusion.
Description: There are two crodws moving against each other. At a point they collide and become a single crowd for a number of frames and in the end they split again.
Sequence: S1.L1.13-57.View1 - 221 frames

In this scenario occlusion happens during 8 frames. Our system is able to chase a crowd even if it is not visible for a certain time. Figs. 6 and 7 depict the behaviour of the tracker: the person marked in yellow circle (left circle with an arrow) gets occluded and then the person is recognized again after she comes out of the bigger crowd.

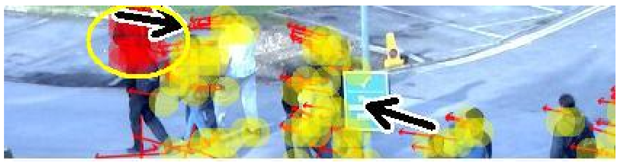

Figure 6: The single person is occluded with the crowd

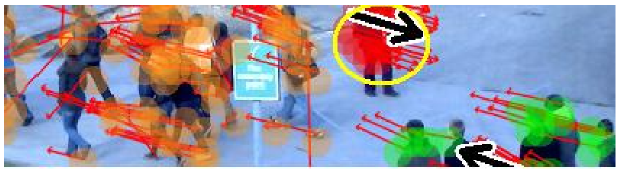

Figure 7: the single person is recognized again after occlusion

4.4 Scenario 4

Challenge type: Arbitraty objects.
Description: There is a car passing by in the front of the camera and it is recognized as a crowd.
Sequence: S1.L1.13-59.View3 - 20 frames

In this scenario we show one of the drawbacks of our system. Since we are not using shape or pattern recognition, the system can classify points belonging to any object as crowds. In the Fig. 8 we can see the car passing by being recognized as a crowd.

4.5 Results

In this section the benchmark results are going to be shown. Mean (μ) and standard deviation(σ) of three variables (Delay, TPR and FN) are shown in table 1. Since $\mu(TPR)$ and $\mu(FN)$ take values 1.0 and 0.0, respectively, their standard deviations are equal to 0. This means that in the above presented scenarios, our system does not detect an event if it

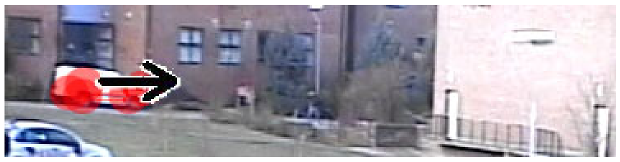

Figure 8: The car is recognized as a crowd.

$\mu(Delay)$	$\sigma(Delay)$	$\mu(TPR)$	$\mu(FN)$
2.66	2.05	1.0	0.0

Table 1: Average results of the system

does not happen and it detects all the events that happen. Regarding the *Delay* we observe that, even though there is some delay in event recognition, it does not affect the overall performances.

5. CONCLUSIONS AND FUTURE WORK

There are several methods to detect and/or track crowds. The usual approaches use excessively computational power and generally focus on a single person or people concentrated in the same space. In many environments, these approaches fail. In this paper, we described an accurate and fast system (20 frames per second) that can effectively detect and track crowds with various shapes. The system has been tested in i386 platform with Windows XP and Vista. It starts with the computation of optical flows, comparing every frame with its previous one, using pyramidal implementation of the Lucas-Kanade method. After applying a block partitioning to each output frame, we cluster the different motion vectors by their similarity, using a DBSCAN algorithm. Finally, to cope with the problem of temporary occlusion, we have introduced a solution: the crowd tracker. Experimental results show that our system is capable of giving an accurate output in real time for different situations and environments. Nevertheless, the distance between motion points is not always accurate, since we lose one dimension when we deal with sequence of images (2D) as input. It can be achieved better results in our system if we provide the 3D coordinates of the motion points.

6. ACKNOWLEDGEMENTS

The authors would like to thank Alessio Gallace, Giuseppe Catalano and Bomi Kim for their contribution to the development of the system.

REFERENCES

[1] T. B. Moeslund, A. Hilton and V. Kruger, ”A survey of advances in vision-based human motion capture and analysis”, *Int. journal of Computer Vision and Image Understanding*, vol. 104, nr. 2-3, pp. 90-126, Nov.-Dec. 2006.

[2] A. M. McIvor, *Background Subtraction Techniques*. Auckland, New Zealand: Reveal Ltd. 2000.

[3] 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, pp. 1337-1342, Oct. 2003.

[4] C. Stauffer and W.E.L. Grimson, ”Adaptive background mixture models for real-time tracking” in *Proc. Conf. Computer Vision and Pattern Recognition*, Ft. Collins, CO, USA, June 23-25. 1999, pp. 246-252.

[5] E.K. Tzamali, M.D. Plumbley and S.A. Velastin, *Motion estimation for crowd analysis using a hierarchical Bayesian approach*. London, UK: CiteSeerX, 2000.

[6] B.A. Boghossian and S.A. Velastin, ”Real time motion detection in video signals”, *High Performance Architectures for Real-Time Image Processing (Ref. No. 1998/197), IEE Colloquium on*, pp. 12/1 - 12/6, Feb. 1998.

[7] B.K.P. Horn and B.G. Schunk, ”Determining optical flow”, *Artificial Intelligence*, vol. 17, pp. 185-203, March 1980.

[8] B.D. Lucas and T. Kanade, ”An iterative image registration technique with an application in stereo vision” in *Proc. Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, Vancouver, Canada, August 24-28. 1981, pp. 121-130.

[9] J. Bouguet, ”Pyramidal implementation of the Lucas-Kanade feature tracker: description of the algorithm”, *Technical report, OpenCV Document*, Intel Microprocessor Research Labs, 2000.

[10] E. Andrade and R. Fisher, ”Modelling crowd scenes for event detection”, *Proc. of the 18th International Conference on Pattern Recognition (ICPR 2006)*, Hong Kong, August 20-24. 2006, pp. 175-178.

[11] B. Zhan, P. Remagnino, N. Monekosso, and S.A. Velastin, ”Self-Organizing Maps for the Automatic Interpretation of Crowd Dynamics”, *Proceedings of the 4th International Symposium on Advances in Visual Computing*, Las Vegas, NV, USA, 2008, pp. 440-449.

[12] Y. Benabbas, N. Ihaddadene, C. Djeraba, ”Global Analysis of Motion Vectors for Event Detection in Crowd Scenes” in *Proc. 11th IEEE International Workshop on PETS*, Miami, USA, June 25. 2009, pp. 109-116.

[13] C. Tomasi and T. Kanade, *Detection and Tracking of Point Features*. Carnegie Mellon University Technical Report CMU-CS-91-132, 1991.

[14] J. Shi and C. Tomasi, ”Good features to track” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, June 21-23. 1994, pp 593-600.

[15] M. Ester et al, ”A density-based algorithm for discovering clusters in large spatial databases with noise” in *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Portland, Oregon, USA, August 2-4. 1996.

[16] A.K. Jain, M.N. Murty and P.J. Flynn, ”Data Clustering: A Review”, *ACM Computing Surveys*, Vol. 31, No. 3, September 1999.

[17] PETS 2009 benchmark data. Available: *http://www.cvg.rdg.ac.uk/PETS2009/a.html*

[18] Crowd analysis video results. Available: *http://kom.aau.dk/~zt/online/CrowdAnalysis/*