Counting Crowded Moving Objects

Vincent Rabaud and Serge Belongie
Department of Computer Science and Engineering
University of California, San Diego
{vrabaud,sjb}@cs.ucsd.edu

Abstract

In its full generality, motion analysis of crowded objects necessitates recognition and segmentation of each moving entity. The difficulty of these tasks increases considerably with occlusions and therefore with crowding. When the objects are constrained to be of the same kind, however, partitioning of densely crowded semi-rigid objects can be accomplished by means of clustering tracked feature points. We base our approach on a highly parallelized version of the KLT tracker in order to process the video into a set of feature trajectories. While such a set of trajectories provides a substrate for motion analysis, their unequal lengths and fragmented nature present difficulties for subsequent processing. To address this, we propose a simple means of spatially and temporally conditioning the trajectories. Given this representation, we integrate it with a learned object descriptor to achieve a segmentation of the constituent motions. We present experimental results for the problem of estimating the number of moving objects in a dense crowd as a function of time.

1 Introduction

This work addresses the problem of segmenting moving objects in video of dense crowds. While our motivating problem is that of counting humans in crowd footage, our approach also has applications to more general groups of objects such as herds of animals or migrating cells. We only require that the crowd be homogeneous, i.e., composed of different instances of the same object class.

The phenomenon of crowding gives rise to a number of challenges. Foremost is the problem of occlusion – both inter-object and self-occlusion. The high incidence of occlusion precludes the use of standard techniques such as blob detection and tracking based on background subtraction. Another problem is that of the large number of independent motions. Existing motion segmentation methods (see for example [20, 24]) are only designed for small numbers of groups (e.g., less than 10) with relatively large, feature-rich regions of support in the image. Frames from a crowd video, in contrast, may contain dozens of erratically moving, feature-impoverished objects. Compounding this problem further is the fact that independent object motions frequently require more than two successive frames to emerge, whereas the majority of motion segmentation methods – both feature based [21] and direct (optical flow based) [8] – only operate on pairs of frames.

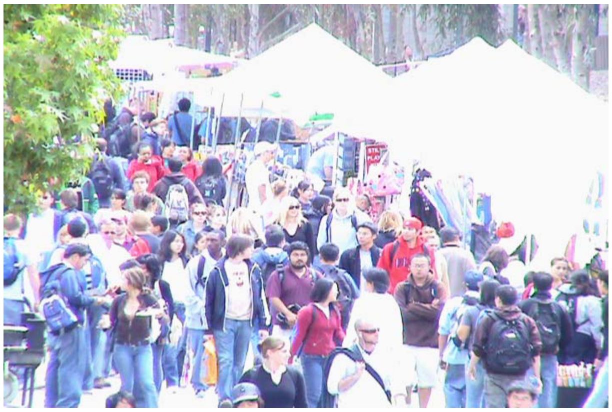


Figure 1. Example of a dense crowd. Our goal is to count the number of moving objects in video sequences such as this.

The approach we propose to address these problems is based on clustering a rich set of extended tracked features and does not require background subtraction. As we make intensive use of low level feature tracking, we have developed a highly parallel version of the KLT tracker [18, 1] which enables us to efficiently populate the spatiotemporal volume with a large set of feature trajectories. An initial pitfall of this step, however, is that the resulting tracks are often fragmented and noisy, thereby making it difficult to cluster them by object. To this end, we propose a conditioning algorithm to smooth and extend the raw feature trajectories. We then cluster the conditioned trajectories into candidate object using local rigidity constraints and a simple object model learned from a small set of training frames labeled only with the ground truth object count.

The organization of this paper is as follows. In Section 2 we discuss related work in the areas of motion segmentation and object tracking. We describe our approach to tracking and segmentation in Section 3. Our experimental results on three different real-world datasets are presented in Section 4. Finally we conclude in Section 5.

2 Related Work

Motion segmentation is a commonly applied preprocessing step for further analyses such as structure from motion, video texture description, robust optical flow computation and object segmentation. For rigid objects, common approaches include two frame RANSAC [19] or, provided that clean, extended feature tracks are available, batch factorization methods [4, 6]. When trying to segment more complex objects, such as articulated ones [27], methods based on local affine invariants [15] show some promise on small numbers of slowly deforming objects.

For the special case of human beings, an abundant literature describes means of finding persons in video footage [16, 23], or even in still frames [25, 11]. Nonetheless, most of these methods have difficulties dealing with dense crowds, though some have proven successful given a dozen persons [28]. Other works take a very different view wherein the crowd is its own entity or even a texture [12], in order to deduce its density [13, 2].

As our motivation is that of counting moving objects, the problem we want to solve is not a conventional tracking problem, even though the solution we propose makes use of a low-level feature tracker. For example, we do not plan to solve tracking through occlusions [7], which frees us from complex manual inputs such as initializations [3, 17] or intricate model definitions [9]. Moreover, the notion of "object" is very subjective as both a crowd and its constituents are objects. Therefore, our algorithm needs to learn *something* about the objects we wish to count.

Finally, the setting of our experiments is that of a surveillance camera viewing a crowd from a distant viewpoint, but zoomed in, such that the effects of perspective are minimized. In a more general setting one can use the calibration method presented in [10]. Contrary to existing methods of counting persons, we do not require overhead views [14], networks of cameras [26] or pre-knowledge of the tracked objects [5].

3 Our Approach

3.1 Feature Tracking

3.1.1 Background

The KLT algorithm [18] is a feature tracking algorithm we choose for its simplicity and practical effectiveness. The driving principle behind the KLT tracker is to determine the motion parameters (e.g., affine or pure translation) of local windows $\mathcal{W}$ from an image I to a consecutive image J. The centers of such windows define the tracked features.

Given a window $\mathcal{W}$, the affine motion parameters A and $\mathbf{d}$ are chosen to minimize the *dissimilarity* :

$$\epsilon = \int_{\mathcal{W}} \left[J(A\mathbf{x} + \mathbf{d}) - I(\mathbf{x})\right]^2 w(\mathbf{x})\mathrm{d}\mathbf{x} \tag{1}$$

where w is a weight function, usually chosen to be constant or Gaussian. It is commonly assumed that only the translation $\mathbf{d}$ matters between two frames, hence leading to the equation:

$$Z\mathbf{d} = \mathbf{e} \tag{2}$$

where

$$Z = \int_{\mathcal{W}} \mathbf{g}(\mathbf{x})\mathbf{g}^\top(\mathbf{x})w(\mathbf{x})\mathrm{d}\mathbf{x}$$

$$\mathbf{e} = \int_{\mathcal{W}} [I(\mathbf{x}) - J(\mathbf{x})]\mathbf{g}(\mathbf{x})w(\mathbf{x})\mathrm{d}\mathbf{x}$$

with

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x}J(\mathbf{x}) \\ \frac{\partial}{\partial y}J(\mathbf{x}) \end{bmatrix}$$

Equation (2) will have a reliable solution if the second moment matrix Z is full rank. Practically, the good windows are chosen to be the ones leading to a Z whose minimal eigenvalue is above a threshold. Once this eigenvalue drops below that threshold, the feature track is terminated.

In its original form, KLT runs until no more initial features can be tracked and avoids features less than ρ pixels apart (we choose $\rho = 4$ pixels).

3.1.2 Increased Efficiency

The KLT tracker requires several parameters, among them, the size of the window. In order for KLT to avail of good features generated by windows of different scales and aspect ratios, we need a meaningful comparisons of dissimilarities corresponding to different windows. We choose to normalize the dissimilarities by the area of their supporting windows. By definition of our parameters, this is equivalent to choosing a uniform weight $w(\mathbf{x}) = \frac{1}{|\mathcal{W}|}$ in Equation 1, where $|\mathcal{W}|$ is the area of the window $\mathcal{W}$.

Consequently, when trying to determine the quality of two different windows $\mathcal{W}$ and $\mathcal{W}'$ centered on a same feature, we simply need to compare the eigenvalues of the normalized matrices $\frac{Z}{|\mathcal{W}|}$ and $\frac{Z'}{|\mathcal{W}'|}$. The resulting simplification is that we only need to associate one window with each feature. The complexity of running KLT is therefore the same as with only one set of windows. We also introduce a speed improvement by using integral images, as used in [22], in order to compute the different matrices Z supported by different window sizes.

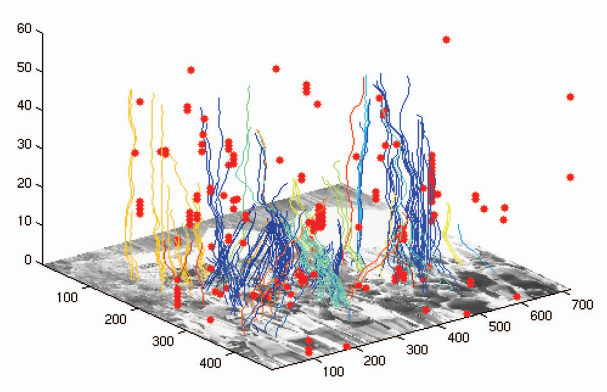


Figure 2. Illustration of feature respawning for the sequence shown in Figure 1. The vertical axis represents time and each colored line represents the trajectory of a feature forward in time. The red dots represent the local maxima of the distance from a point to the closest trajectory, and hence the centers of the "holes" among the trajectories.

Another important aspect of this approach is that when tracking a crowd of similar objects, our version of KLT can run through sample training frames first, in order to determine the parameters leading to the best windows. Later, during execution time, only the optimal sets of parameters need to be used. Those usually represent less than 5% of the possible parameter sets. This is an important step as the optimal window sizes depend on the object class; for example a weakly textured object requires bigger windows.

3.1.3 Feature Re-Spawning

After some time, KLT loses track of features for the following reasons: inter-object occlusion, self-occlusion, exit from the picture, and appearance change due to perspective and articulation. Therefore, new features need to be created.

The common implementation of KLT simply re-creates features at every frame. This can be computationally intensive and inefficient because weak features can be renewed and tracked, despite being uninformative. We propose to respawn features only at specific locations in space and time and then propagate them forward and backward in time.

If we instantiate the tracker at a frame and allow it to run, the spatio-temporal volume fills with tracks of features as shown in figure 2. The populated space contains empty regions which are the ideal places for re-spawning features. However, for computational reasons, we cannot recreate features in each of these holes; moreover, some holes are obviously bigger and consequently require more attention.

We propose the following approach: each hole center is given a weight corresponding to its distance to the closest neighboring trajectory. The frame at which we should respawn features is simply the weighted average of the times at which the centers are located. Figure 2 contains an example with 50 frames, and the best frame for re-spawning features occurs at time 20.

3.2 Trajectory Conditioning

In practice, the KLT tracker gives us a set of trajectories with poor homogeneity: the trajectories do not necessarily end and begin at the same times, occlusions can result in trajectory fragmentation, and a feature can lose its strength resulting in less precise tracks. In order to have an improved set of features, we condition the data, both spatially and temporally.

We consider that even though each trajectory was computed independently, it is influenced by its spatial neighbors. The algorithm described in Figure 3 considers the propagation of a box[1] along a trajectory and it is applied to each raw trajectory in the video sequence. Initialization is performed by computing the centroid of the coordinates of all the trajectories present inside the box at the time the trajectory begins.

This algorithm is slightly more complicated than the propagation of an averaging kernel along the trajectory because the smoothing occurs even after the original trajectory is dead. The resulting output is therefore a set of smoother, extrapolated and less fragmented trajectories.

3.3 Trajectory Clustering

In order to determine the number of objects at a given time t, the features present at that time are going to be clustered into plausible objects, and the number of resulting clusters will simply give the answer. Even though an instant t is considered, the whole life of the features has to be considered for partitioning: two objects might be close at one instant, hence looking like one entity, but might part later, resulting in the impossibility for their features to be grouped together. We will therefore focus our attention on a time window centered on t (we chose a half-width of 200 frames in our implementation).

At each time step, the present features form the nodes of a connectivity graph $\mathcal{G}$, whose edges indicate *possible* membership to a common object. The problem is now an instance of graph partitioning with binary edge weights that can be solved using cues and techniques we now describe.

[1] The diameter of the neighborhood box is chosen to be 4ρ, with ρ defined at the very end of Section 3.1.1. This decision is motivated by the following: trajectories cannot be closer than ρ by definition. Therefore, choose the next order, 2ρ as a radius of proximity, hence 4ρ as a diameter.

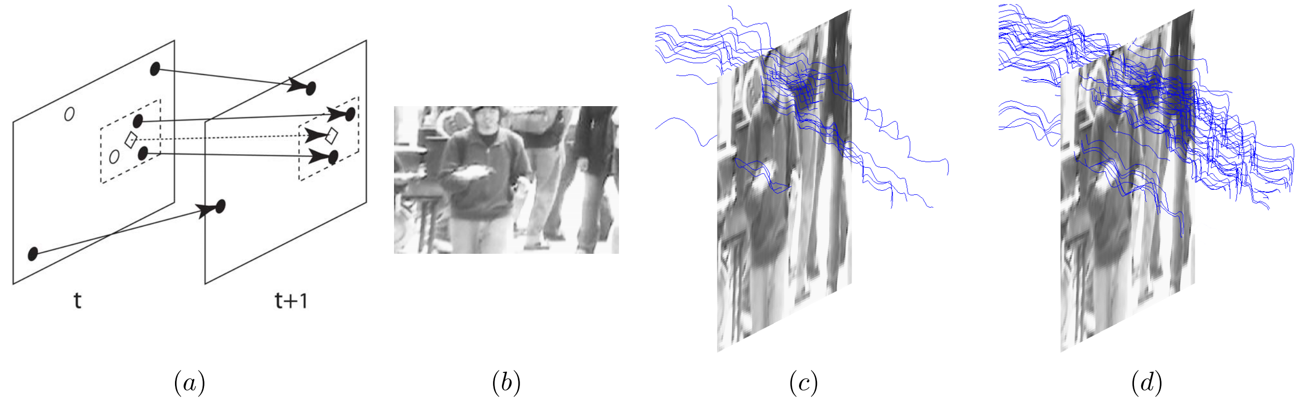


Figure 3. (a) Illustration of conditioned trajectory at time t with coordinates indicated by diamond symbol. Empty circles indicate feature trajectories that do not continue past t, while filled circles continue to time $t+1$ according to the depicted individual displacement vectors. The next coordinate of the conditioned trajectory is computed via the displacement vector, shown as a dotted arrow, obtained by averaging the vectors inside the box centered on the diamond. This process is carried out for all trajectories in the sequence, and is run forward and backward in time. The result is a set of longer, smoother trajectories with reduced fragmentation. (b) shows an example of a detail of a crowd sequence on which features are tracked (c) and then conditioned (d).

3.3.1 Connectivity Graph

The set of trajectories we are dealing with is so heterogeneous in length, type and amount of overlap that common approaches, like the shape interaction matrix [4], are unusable for clustering. This section demonstrates how cues on the shape can prune many connections in our connectivity graph $\mathcal{G}$, hence simplifying its partitioning.

We will use the following notations: $\boldsymbol{\tau}^i$ refers to the track of feature i and $\boldsymbol{\tau}^i(t)$ refers to the homogeneous coordinates $(\boldsymbol{\tau}^i_x(t), \boldsymbol{\tau}^i_y(t), 1)^\top$ of the feature i at time t. Finally, $\|.\|_2$ refers to the L_2 norm.

First, let us assume a *bounding box* of the objects in our crowd is known: a box, as small as possible, able to contain every possible instance of the object. Formally, the width and height of this box are defined as follows:

$$w = \max_{(\boldsymbol{\tau}^i,\boldsymbol{\tau}^j)\in \text{Object}^2} \left(\max_{t\in \text{Time}} |\boldsymbol{\tau}^i_x(t) - \boldsymbol{\tau}^j_x(t)| \right)$$
$$h = \max_{(\boldsymbol{\tau}^i,\boldsymbol{\tau}^j)\in \text{Object}^2} \left(\max_{t\in \text{Time}} |\boldsymbol{\tau}^i_y(t) - \boldsymbol{\tau}^j_y(t)| \right)$$

By definition, two trajectories that cannot be put into such a box at one point of their lifetime cannot belong to the same object.

Similarly, let us define the *articulation factor* for an object:

$$\phi = \max_{(\boldsymbol{\tau}^i,\boldsymbol{\tau}^j)\in \text{Object}^2} \left(\max_{t\in \text{Time}} \|\boldsymbol{\tau}^i(t) - \boldsymbol{\tau}^j(t)\|_2 - \min_{t\in \text{Time}} \|\boldsymbol{\tau}^i(t) - \boldsymbol{\tau}^j(t)\|_2 \right)$$

It simply defines the variation through time of the distance between two features, hence how “loose” their connection is. This factor depends on the object kind as well as its movement: for example, for a rigid object under pure translation, $\phi = 0$.

By definition of these three parameters, if two trajectories $\boldsymbol{\tau}^i$ and $\boldsymbol{\tau}^j$ do not comply to one of the three conditions:

$$\begin{gathered} \forall t, |\boldsymbol{\tau}^i_x(t) - \boldsymbol{\tau}^j_x(t)| \leq w \\ \forall t, |\boldsymbol{\tau}^i_y(t) - \boldsymbol{\tau}^j_y(t)| \leq h \\ \max_{t\in \text{Time}} \|\boldsymbol{\tau}^i(t) - \boldsymbol{\tau}^j(t)\|_2 - \min_{t\in \text{Time}} \|\boldsymbol{\tau}^i(t) - \boldsymbol{\tau}^j(t)\|_2 \leq \phi \end{gathered} \tag{3}$$

then, these two trajectories do not belong to the same object. Therefore, the conditions in Equation (3) are *necessary* for grouping $\boldsymbol{\tau}^i$ and $\boldsymbol{\tau}^j$ together.

We will detail in Section 4 how we obtain these parameters (w, h, ϕ).

3.3.2 RANSAC-based Merging

While we do not assume a prior on the objects, assuming that they behave rigidly in some of their parts/limbs can lead to an initial grouping.

If several features share a rigid motion in 3D, then the motion of their 2D orthographic projections is affine. Therefore, we assume that if 2D features share an affine movement during their whole life span, they then belong to a rigid part of an object, and consequently to a common object. Mathematically, we assume that a set $\mathcal{T}$ of features $\boldsymbol{\tau}^i$

has a 3D rigid motion if it exists a family of 3×3 homogeneous affine motion matrix $M_{t \to t'}$ such that $\forall \boldsymbol{\tau}^i \in \mathcal{T}$ and $\forall (t, t') \in \text{Time}^2$:

$$\|M_{t \to t'} \cdot \boldsymbol{\tau}^i(t) - \boldsymbol{\tau}^i(t')\|_2 = 0 \qquad (4)$$

Nonetheless, because of the noise present during the KLT tracking, this assumption needs to be relaxed. In Section 3.1.1, ρ was defined as the minimum distance between two features but it can also be interpreted as the distance below which ambiguity disappears between features. The condition in Equation (4) can now be relaxed as the following *sufficient* condition for grouping trajectories in a set $\mathcal{T}$:

$$\|M_{t \to t'} \cdot \boldsymbol{\tau}^i(t) - \boldsymbol{\tau}^i(t')\|_2 \leq \rho \qquad (5)$$

To determine these rigid clumps $\mathcal{T}$, RANSAC is applied to sets of trajectories existing on the studied time window and connected in the connectivity graph $\mathcal{G}$. By considering triadic connections, the corresponding affine motions through time are computed via least square minimization and tested with potential inliers on Equation (5).

At this point of the feature clustering, we have clumps containing one or several features belonging to a common rigid motion, as well as possible connections between features, defined in the connectivity graph $\mathcal{G}$. Empirically, we found that the number of clumps is usually between 20% and 40% of the number of features.

3.3.3 Agglomerative Clustering

Finally, the features are agglomeratively clustered into independent objects. At each iteration, the two closest sets $\mathcal{T}$ and $\mathcal{T}'$ are considered and if all their features are linked to each other in the connectivity graph $\mathcal{G}$, they are merged together. Otherwise, the next closest sets are considered. We proceed until all the possible pairs have been analyzed.

4 Experiments

4.1 Implementation

We implemented our algorithm in C++ for the KLT part, and Matlab for the clustering side. We name it ALDENTE, as it is an ALgorithm for DEtermining the Number of Trackable Entities.

A final point we need to clarify is how we obtain the *bounding box* parameters w and h as well as the *articulation factor* ϕ. For the sake of simplicity, we only want the training data of the algorithm to be as simple as a video footage associated with its ground truth number of objects through time. Therefore, when given the training data, the algorithm simply sweeps though the possible sets (w, h, ϕ) and keeps the one leading to the minimal error when executing our tracking/clustering approach.

	USC	LIBRARY	CELLS
# frames	900	1000	200
characteristics	320×240, MPEG, 15 fps	640×480, DV, 30 fps	320×240, JPEG seq.
KLT time	1 min	6 min	4 min
Clustering time	20s	42s	16s
# features/frame before conditioning	60	602	610
# features/frame after conditioning	67	737	717
Feature duration before conditioning	138	128	24
Feature duration after conditioning	160	162	52
Optimal (w, h, ϕ)	50,60,50	100,180,45	50,50,20
Average error (#)	0.8	2.7	24
Average error (%)	10	6.3	22

Figure 4. Summary of the results

4.2 Datasets

Our approach was tested on three different datasets:

1. USC. The dataset from [28]: an elevated view of a crowd consisting of zero to twelve persons.
2. LIBRARY. Our dataset consists of an elevated view of a crowd of twenty to fifty persons. The background is never completely visible.
3. CELLS. A red blood cell dataset: fifty to a hundred blood cells are moving at different speeds. The cells are blobs changing shapes through time, sometimes occluding each other and having different speeds.

The ground truth for the number of objects has been determined by several human operator at certain frames of the data, by looking at the previous and next frame to determine how many objects moved.

Table 4 summarizes some characteristics of this data and obtained results. Figures 5,6,7,8 show a summary of the analysis through time as well as some close-ups on certain results

4.3 Analysis

As shown in Table 4, conditioning increases the life of the features and hence the average number of features present in a frame. An interesting result appears with the optimal sets of parameters (w, h, ϕ). The found optimal sets are close to what is expected: the *bounding box* parameters have dimensions very close to the size of the objects we count (e.g. vertical box for the persons, and square ones for the cells).

The cell dataset presented difficulties not so much because of the lack of good features to track but because of the poor frame rate leading to some untrackable cells. We nonetheless captured the global variation in the number of cells, as shown in Figure 5c.

5 Discussion and Conclusion

Multi-body motion segmentation algorithms have relied thus far on certain assumptions made on the scene, on the objects, or on their number. In an attempt to free these common approaches from complicated parameter tuning, we experimented a new way of segmenting motions generated by multiple instances of an object in a crowd.

We introduced some enhancements to the KLT tracker in order to extract a large set of features from the video footage. After proposing a conditioning technique for feature trajectories, we introduced a trajectory set clustering method to identify the number of moving objects in a scene.

With respect to the encouraging results we obtained, we propose to extend our method in order to identify a more complex model, in appearance and motion, of the objects. We also plan to investigate combining our approach with static object counting methods. Further improvement will include autocalibration (at least to correct the perspective) and discrimination of the background from the objects, in order to have the method work for handheld camera.

Acknowledgments

This work was partially supported under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48., NSF CAREER #0448615, the Alfred P. Sloan Research Fellowship and the UCSD division of the California Institute for Telecommunications and Information Technology, Calit2. The authors would also like to thank S. Agarwal and B. Ochoa for helpful discussions.

References

[1] S. Birchfield. KLT: An implementation of the kanade-lucas-tomasi feature tracker. http://vision.stanford.edu/birch/klt/.

[2] B. A. Boghossian and S. A. Velastin. Evaluation of motion based algorithms for automated crowd management. In A. Clark and P. Courtney, editors, *Workshop on Performance Characterisation and Benchmarking of Vision Systems*, Canary Islands Spain, Jan 1999.

[3] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *ICCV99*, pages 1197–1203, 1999.

[4] J. Costeira and T. Kanade. A multibody factorization method for independently moving-objects. *IJCV*, 29(3):159–179, September 1998.

[5] Y. Do and T. Kanade. Counting people from image sequences. In *Int. Conf. Imaging Science, Systems, and Technology*, 2000.

[6] A. Fitzgibbon and A. Zisserman. Multibody structure and motion: 3-d reconstruction of independently moving objects. In *ECCV00*, pages I: 891–906, 2000.

[7] Y. Huang and I. Essa. Tracking multiple objects through occlusions. In *CVPR*, June 2005.

[8] M. Irani and P. Anandan. All about direct methods. In *International Workshop on Vision Algorithms*, pages 267–277, 1999.

[9] M. Isard and J. MacCormick. Bramble: A bayesian multiple-blob tracker. In *ICCV01*, pages II: 34–41, 2001.

[10] N. Krahnstoever and P. R. S. Mendona. Bayesian autocalibration for surveillance. In *ICCV*, 2005.

[11] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *CVPR05*, pages I: 878–885, 2005.

[12] A. Marana, S. Velastin, L. Costa, and R. Lotufo. Automatic estimation of crowd density using texture. *Safety Science*, 28(3):165–175, apr 1998.

[13] N. Paragios and V. Ramesh. A MRF-based approach for real-time subway monitoring. In *CVPR01*, pages I:1034–1040, 2001.

[14] A. Schofield, P. Mehta, and T. Stonham. A system for counting people in video images using neural networks to identify the background scene. *PR*, 29(8):1421–1428, August 1996.

[15] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. In *ECCV04*, pages Vol II: 85–98, 2004.

[16] Y. Song, X. Feng, and P. Perona. Towards detection of human motion. In *CVPR00*, pages I: 810–817, 2000.

[17] F. Tang and H. Tao. Object tracking with dynamic feature graphs. *VS-PETS*, October 2005.

[18] C. Tomasi and J. Shi. Good features to track. In *CVPR94*, pages 593–600, 1994.

[19] P. Torr. Motion segmentation and outlier detection. In *Oxford Univ.*, page Ouel 1987/93, 1987.

[20] P. H. S. Torr. Geometric motion segmentation and model selection. In J. Lasenby, A. Zisserman, R. Cipolla, and H. Longuet-Higgins, editors, *Philosophical Transactions of the Royal Society A*, pages 1321–1340. Roy Soc, 1998.

[21] P. H. S. Torr and A. Zisserman. Feature based methods for structure and motion estimation. In *Proceedings of the International Workshop on Vision Algorithms*, pages 278–294. Springer-Verlag, 2000.

[22] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, May 2004.

[23] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *IJCV*, 63(2):153–161, July 2005.

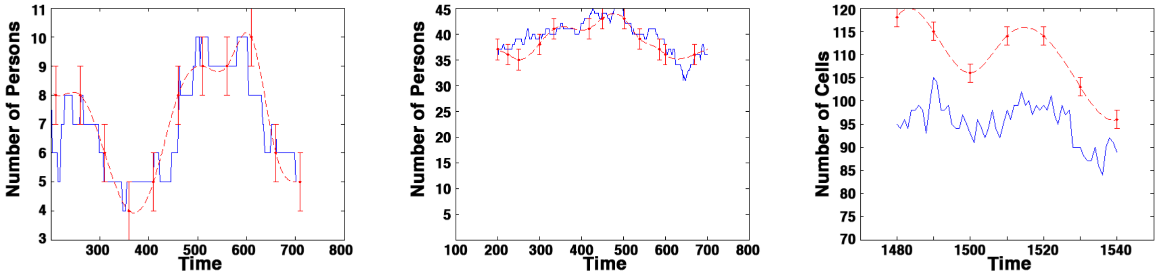
[24] J. Wills, S. Agarwal, and S. Belongie. What went where. In *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, volume 1, pages 37–44, 2003.

[25] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *ICCV05*, pages I: 90–97, 2005.

[26] D. Yang and H. Gonzalez-Banos. Counting people in crowds with a real-time network of simple image sensors. In *ICCV03*, pages 122–129, 2003.

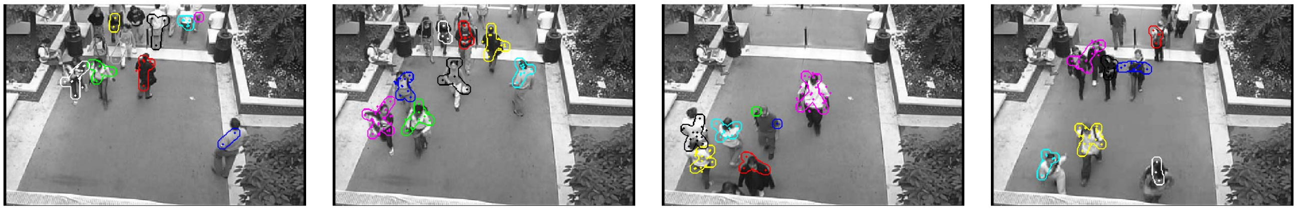
[27] L. Zelnik-Manor and M. Irani. Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations. In *CVPR03*, pages II: 287–293, 2003.

[28] T. Zhao and R. Nevatia. Bayesian human segmentation in crowded situations. In *CVPR03*, pages II: 459–466, 2003.



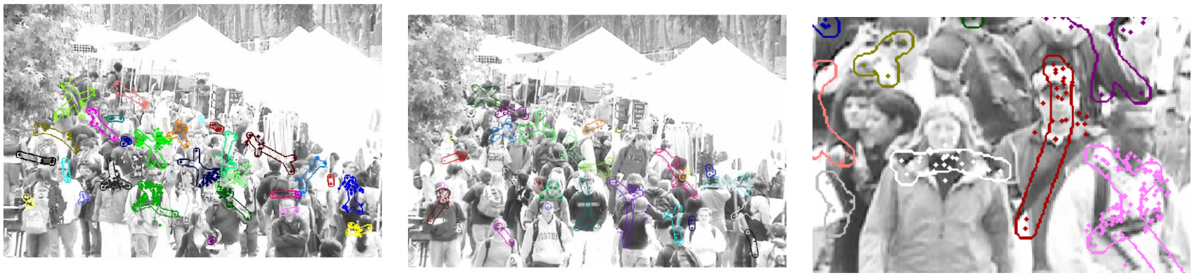
(a) (b) (c)

Figure 5. Number of objects through time estimated by our algorithm (solid blue) compared to ground truth (dashed red). Error bars indicate one standard deviation of the ground truth labels determined by several human operators. (a) USC (b) LIBRARY (c) CELLS.



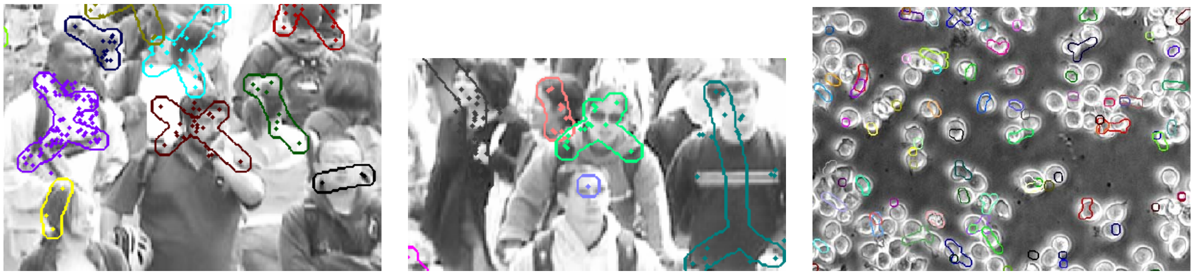
(a) (b) (c) (d)

Figure 6. Results of clustering on the USC dataset: (a) and (b) show good performances on several persons while sometimes persons are merged as shown in (c) and (d).



(a) (b) (c)

Figure 7. Results of clustering on the LIBRARY dataset shown in (a) and (b). (c) shows a close-up with good clustering except for two persons on the left that were standing up and not exhibiting any motion.



(a) (b) (c)

Figure 8. Additional examples of clustering results on the LIBRARY (a,b) and CELLS dataset (c).