Real-time, Robust Target Tracking in Videos via Graph-cuts

Barak Fishbain[a], Dorit S. Hochbaum[b] and Yan T. Yang[b]

[a]Department of Environmental, Water and Agricultural Engineering, Faculty of Civil & Environmental Engineering, Technion - Israel Institute of Technology Haifa, Israel;
[b]Department of Industrial Engineering and Operations Research, University of California, Berkeley, California, USA

ABSTRACT

Video tracking is a fundamental problem in computer vision with many applications. The goal of video tracking is to isolate a target object from its background across a sequence of frames. Tracking is inherently a three dimensional problem in that it incorporates the time dimension. As such, the computational efficiency of video segmentation is a major challenge. In this paper we present a generic and robust graph-theory-based tracking scheme in videos. Unlike previous graph-based tracking methods, the suggested approach treats motion as a pixel's property (like color or position) rather than as consistency constraints (i.e., the location of the object in the current frame is constrained to appear around its location in the previous frame shifted by the estimated motion) and solves the tracking problem optimally (i.e., neither heuristics nor approximations are applied). The suggested scheme is so robust that it allows for incorporating the computationally cheaper MPEG-4 motion estimation schemes. Although block matching techniques generate noisy and coarse motion fields, their use allows faster computation times as broad variety of off-the-shelf software and hardware components that specialize in performing this task are available. The evaluation of the method on standard and non-standard benchmark videos shows that the suggested tracking algorithm can support a fast and accurate video tracking, thus making it amenable to real-time applications.

Keywords: Surveillance, Target Tracking, Network Flow Algorithms, Motion estimation, Video Compression, MPEG-4

1. INTRODUCTION

Target tracking is the process of delineating a target object from its background across a sequence of frames. The tracking problem is three dimensional in that it incorporates the time dimension. As such, the computational efficiency of any suggested solution is a major challenge. The method presented here is efficient enough to process videos under near real-time constraints.

Tracking algorithms in the literature are categorized into three main classes. The first class includes variational motion segmentation with level sets,[1] and fast geodesic active contour method.[2] At heart of this variational computation approach is the use of *continuous* models. However, digital videos are innately *discrete*. The conversion of these real-numbers solutions to discrete ones is not straight forward and often requires heuristics and further processing. The second class of tracking techniques incorporates statistical and stochastic elements.[3–5] These statistical schemes rely heavily on iterative steps that are computationally intense and do not guarantee optimal solution nor consistency (i.e., the same output over sequential runs on the same input data). The third approach, on which we focus here, formulates the problem as a graph cut problem. The use of graph cuts for object tracking was first introduced by Xu et al.,[6] where the tracked object's contour in frame t was sought in a narrow region in frame $t+1$. This method did not utilize motion information and therefore faced difficulties when dealing with large displacements and occlusions. A few graph-cuts based tracking algorithms that utilize motion data were reported.[7–10] Freedman and Turek,[7] suggested a two-phase tracking mechanism.

Further author information: (Send correspondence to B.F)
B.F: E-mail: fishbain@technion.ac.il, Telephone: +972-4-829-3177
D.S.H: E-mail: hochbaum@ieor.berkeley.edu, Telephone: +1-510-642-4998
Y.T.Y: E-mail: yxy128@berkeley.edu

Real-Time Image and Video Processing 2013, edited by Nasser Kehtarnavaz, Matthias F. Carlsohn, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 8656, 865602 · © 2013 · SPIE-IS&T
CCC code: 0277-786X/13/$18 · doi: 10.1117/12.2002947

At the first stage the motion in the sequence is extracted by graph-cuts based optical flow. Then all the motion vectors are grouped into spatio-temporal blobs, each representing a moving object. The grouping process is done through propagation, hence if pixel $I_{x,y,t}$ was found to move to location $(x\prime, y\prime)$ at frame $t+1$, than both pixels $(I_{x,t,y}, I_{x\prime,y\prime,t+1})$ are grouped into one object. This scheme, however, does not guarantee that the tracked objects do not break into several small components within a few frames. Malcolm et. al[8] used an autoregressive model to provide a prediction of the target's location in the succeeding frames. This prediction is then used to construct the spatial constraints on the object's expected location in these frames. This algorithm, however, does not handle occlusions well, since it does not introduce a specific process for dealing with interacting objects (see[10] for such example). Bugeau and Pérez[9] utilized Lucas and Kanade's optical flow algorithm[11] in a two-phase tracking mechanism. At the first stage all objects are tracked individually. Then at the second stage objects that might have been merged in the first phase are segmented. An extension of the latter work that addresses occluded objects was introduced by Papadakis and Bugeau.[10]

A major drawback of all aforementioned video tracking schemes that utilize motion data, is the use for motion estimation of *optical-flow* methods.[11,12] While considered to be most accurate, these optical-flow methods require the minimization of an energy functional. In order to solve the resulting large sparse systems numerically, classical iterative methods are commonly used. While these are simple to implement, their convergence is slow, and often thousands of iterations are necessary to get sufficiently close to the global minimum of the energy functional. This is the reason why optical flow methods are slow and unsuitable for time-critical applications.

In this paper we suggest a generic robust graph-theory-based tracking scheme in videos. The suggested method casts the tracking problem as a variant of the normalized cut (NC') problem.[13] This approach is unique in that it treats motion as pixel's features (like color or position). This is in contrast to the previously suggested methods,[7–10] which presented motion as consistency constraints. Thus, the location of the object in the current frame is constrained to appear around its location in the previous frame shifted by the estimated motion. Because of that, there is no need for the heuristics commonly used in dealing with difficulties associated with this type of constraints. Similar notions can be found in human action recognition algorithms, where the similarity between nodes is measured either by using descriptors[14] or by the motion field computed by optical flow.[15]

The suggested scheme is so robust that it allows for incorporating the computationally cheaper Moving Picture Experts Group (Rev. 4), MPEG-4, block-matching, motion estimation schemes.[16] Although block matching techniques generate noisy and coarse motion fields, their use here has two advantages: (i) Faster computation times as broad variety of off-the-shelf software and hardware components that specialize in performing this task and can easily be incorporated into the segmentation scheme are available; and (ii) If the videos are already compressed, then the motion information is inherent in their compressed form, and is available from the video encoder. In that case there is no need to apply *any* motion estimation algorithm. This approach of using the motion field coded within the compressed sequence was previously suggested for video enhancement.[17,18] Graph-based object detection and tracking in H.264/AVC bitstreams was recently suggested by Sabrin et. al.[19] However the graph there is used only to build the association of the data. No graph-based algorithms, which could have enhanced the performance of the algorithm, were exploited for the task.

Consequently, the contribution of this paper is two-fold: Firstly, it formulates the tracking problem as a graph problem; secondly, it demonstrates that the graph-theory-based tracking scheme developed here is robust enough, allowing using coarse and noisy block matching motion fields in the process. The results here demonstrate that our scheme can support a fast and accurate video tracking, which make the suggested scheme a perfect choice for many tracking in video applications.

The paper is organized as follows: Section 2 formulates the tracking problem as NC' problem. Section 3 addresses practical aspects of the system; Section 4 presents a performance evaluation of the algorithm on real-life benchmark videos; and Section 5 provides concluding remarks.

2. PROBLEM FORMULATION

2.1 A graph representation of video tracking

Target tracking is presented as a bi-partitioning problem in a graph representing the video, where one set of the bi-partition represents the tracked object and through this tracking is achieved. Specifically, the problem is presented on an undirected graph $G = (V, E)$ with a set of nodes V, representing pixels in their spatiotemporal position, and a set of edges E, connecting each node to its adjacent pixels. For videos, one typically considers three dimensional graphs with pixels arranged along a grid. The 6-neighbors set up is a commonly used adjacency rule with each pixel having 6 neighbors – two along the vertical axis, two along the horizontal axis and two along the temporal axis. The 26-neighbors arrangement, which includes the adjacent pixels along the diagonal axes is also a common setup. We use here a 10-neighborhood model: Each pixel has a total of 10 neighbors: 4 in the current frame (up, down, left and right) and three additional neighbors in the pixel's corresponding locations in the 3 preceding and 3 subsequent frames. This is illustrated in Figure 1. The similarity is computed for each pair of neighboring pixels. All edges between non-neighboring pixels are assigned zero weights.

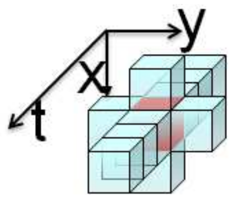

Figure 1. A pixel and its spatio-temporal neighborhood.

The edges in the graph carry similarity weights. This similarity may take into account multiple pixels' features such as the pixel's neighborhood texture, its intensity, corresponding motion and its color or brightness. In terms of the graph, each edge $[i, j]$ is assigned a similarity weight w_{ij} that increases as the two pixels i and j are perceived to be more similar. Low values of w_{ij} are interpreted as *dissimilarity.*

The following notation facilitates the presentation of the tracking optimization problem: A bi-partition of the graph is called a *cut*, $(S, \bar{S}) = \{[i, j] | i \in S, j \in \bar{S}\}$, where $\bar{S} = V \setminus S$. We denote the *capacity of a cut* $(S, \bar{S})$ as:

$$C(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}, [i,j] \in E} w_{ij}. \tag{1a}$$

The *capacity of a set* $S \subset V$ is denoted by:

$$C(S, S) = \sum_{i,j \in S, [i,j] \in E} w_{ij}. \tag{1b}$$

We denote the *volume of a set*, the sum of all edges connected to at least one node in a set, S, by:

$$d(S) = C(S, V) = \sum_{i \in S, j \in V, [i,j] \in E} w_{ij} \tag{1c}$$

Using the notation above, the tracking problem is cast as finding a bi-partition, $\left(S, \bar{S}\right)$, that minimizes a ratio of two objectives, one has S maximize the similarity of the pixels within the group and the second goal is to minimize the similarity between S and its complement ($\bar{S} = V \setminus S$). This can be written as the ratio[13] :

$$\min_{S \subset V} \frac{C(S, \bar{S})}{C(S, S)}. \tag{2}$$

Hochbaum showed[13] that (2) is equivalent to minimizing one term in Shi's and Malik's *Normalized Cut* (NC) optimization criterion.[20] Consequentially we refer to this problem as a *variant of normalized cut* or NC′ in short. Hochbaum has also showed,[13] that (2) is solvable in polynomial time and that this optimization criterion is efficient and extremely robust for image segmentation.

3. IMPLEMENTATION CONSIDERATIONS

3.1 Similarity Measures

The system's input consists of two *vectors*: $\vec{I}_{klt}$ and $\vec{m}_{klt}$. $\vec{I}_{klt}$ is the color representation vector of the pixel coordinates (k, l) of frame t. The color representation can be in any form (e.g, R-G-B, Y-Cb-Cr, H-S-V or L-a-b). The vector $\vec{m}_{klt}$ is the motion component which typically contains two components: the horizontal and vertical motions. For the subsequent processing stages, the translation vector is presented in polar coordinates, hence magnitude, A_{klt}, and angle, φ_{klt}, of the motion vector.

We incorporate these two vectors for each pixel in a *feature vector*, $\vec{F}$, consisting of 5 parameters - 3 for color representation and two for motion. These features quantify the resemblance between pairs of pixels. To this end, several quantifiers can be used to measure the similarity: correlation, t statistical test and L^1 or L^2 norms. Here we use the L^2 norm as a measure of dissimilarity: the larger the norm the greater the difference between the two pixels. Consequently, the reciprocal of this quantity is a measure of similarity between two pixels i and j,

$$w_{ij} = 1/(\|\vec{F}_i - \vec{F}_j\|_2 + \epsilon). \tag{3}$$

3.2 Block-Matching-Based Motion Estimation Techniques

The concept behind block matching motion estimation is to divide the current frame into a matrix of macro-blocks. The translation vector of each of these blocks is estimated by searching the most similar block in the preceding frame. The matching is based on the output of a cost function. The location in the previous frame that results in the least cost is the one that matches best the current block. There are various cost functions, of which the most popular and least computationally expensive is the sum of absolute difference (SAD). Another common cost function is the mean squared error (MSE).

Several block-matching high efficiency algorithms were presented[16, 21, 22] . By applying certain assumptions on the error function, such as smoothness and global minima, these methods reduce the computational complexity: The number of possible matching candidate blocks, examined within the entire previous frame or within a bounded search area, is reduced by using efficient location patterns for candidate blocks, such as diamond or spiral; and by introducing maximum desirable error value, an early-stopping criterion is applied. These improvements are traded off with possible degradation in motion estimation accuracy and the presence of noise in the computed motion field. The degradation is substantiated by the tremendous reduction in running times. Specifically, we use here the $x.264$[23] implementation of *diamond search* motion estimation algorithm[16] , which is commonly used in MPEG-4 video compression standard.

Figure 2 illustrates the block matching motion field computed by diamond search[16] for two sequences, one sequence taken from the CAVIAR data set[24] and a sequence of the New York Stock Exchange's facade.[25] Figure (a) shows a representing frame from the CAVIAR sequence. Figure (b) presents the corresponding motion field's amlitudes. Figure (d) presents a blowup of the motion field of the small segment marked on Figure (c). Both examples clearly show that the motion fields, generated by the block matching diamond search motion estimation technique, are coarse and noisy. In spite of these characteristics of the motion field, the tracking scheme presented is robust and manages to utilize the motion field for the tracking task. This results in a computationally efficient mechanism as both the computation of the motion field and the tracking realization are extremely efficient.

3.3 Seed Nodes

The target of interest to be tracked is not always the salient nor the only feature in the frame. In order to specify the object of interest, one or more pixels are a priori assigned as foreground or background. These pixels correspond to the *seed nodes* in the graph. Seeds may be selected with either a manual or automatic procedure. It is possible to run the NC' segmentation with a single foreground and a single background nodes. In cases where the segmentation criterion, (2), results in an unsatisfactory results, such as a very large $|S|$, one can add few more nodes (by clicking on relevant pixels in the sequence), which often results in a significant improvement. This simulates the course of action of a human operator, where the target of interest is indicated in the first few frames by the operator's mouse clicks and then the algorithm delineates and tracks the object in all subsequent frames.

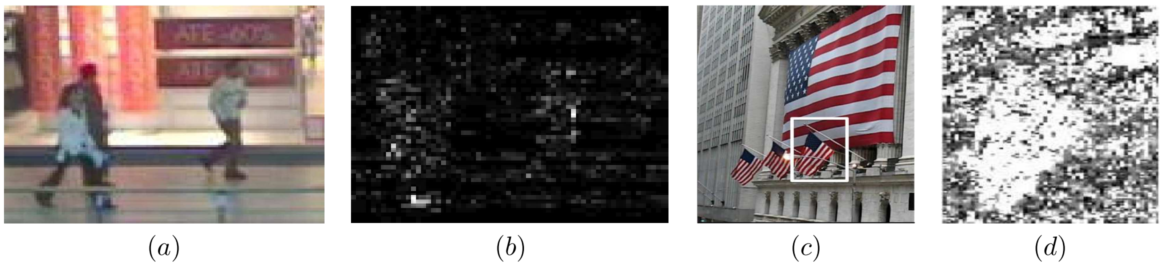

Figure 2. New-York Stock Exchange facade sequence (a) and the motion amplitudes (b) of the flag fragment in (a), brighter pixels correspond to larger amplitudes. Figure (c) is a frame from a surveillance sequence extracted from the CAVIAR data set[24] and (d) presents its corresponding motion field

3.4 Segmentation over Long Image Sequences

The method described here takes in a fixed number of image frames, and processes this batch. In this way the algorithm can incorporate information across several frames to produce the best partition. When considering long image sequences the algorithm processes the sequence in a temporal moving window fashion, where N frames are processed at each window's location. The process is described in Figure 3. As described in Section 3.3, few nodes are a priori tagged as foreground or background (seed nodes). After the required seed nodes are indicated, a window of N frames is processed. Then the tracking results of the last frame in the first batch are used as seed nodes for the segmentation of the next $N-1$ frames. This process is repeated till all frames are processed. This mode of operation is prone to error propagation over time. This can be compensated by additional user inputs in any window's position. It is important to note that while additional user input throughout the process may improve the tracking results with no computational cost, the user's input is required only at the beginning of the process for identifying the target of interest. Our experiments show that a window size of $N = 10$ was a good tradeoff between computation time and accuracy. Following the discussion in Section 2.1, pixel's neighborhood is defined over 7 frames. If $N < 7$, then the pixel's neighborhood is truncated symmetrically around the central pixel.

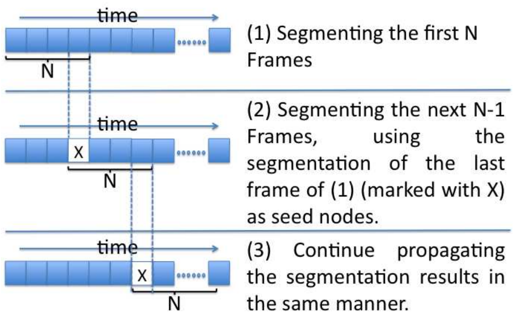
time
N
(1) Segmenting the first N Frames
time
X
N
(2) Segmenting the next N-1 Frames, using the segmentation of the last frame of (1) (marked with X) as seed nodes.
time
X
N
(3) Continue propagating the segmentation results in the same manner.

Figure 3. Segmenting Long Video Sequences by propagating the segmentation results over consecutive moving window positions

4. EXPERIMENTAL RESULTS

In order to solve the NC$'$ problem we use the Hochbaum's PseudoFlow (*HPF*) algorithm.[26] The HPF algorithm has a strongly polynomial complexity and it was found to outperform any other solution approaches in general,[27] and for vision problems in particular.[28] The output of HPF is a bi-partition that divides the spatiotemporal pixels into two groups: one group is the delineated target object, and the other corresponds to background. HPF implementation was downloaded from.[29]

4.1 Data Sets and Performance Measures

The suggested method was tested on a broad variety of standard and non-standard test scenarios. Standard test scenarios sequences were taken from: the Context Aware Vision using Image-based Active Recognition (CAVIAR) database;[24] the HDTV TRICTRAC test sequences;[30] and the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) benchmark scenarios.[31]

4.2 Results

The tracking of the flag (marked with a square in Figure 2(a)) in the NYSE sequence, which is characterized by heavy global motion, is done with a reduced feature space: As color information, we use the pixels' intensity levels; while for motion we use only motion amplitudes (shown in Figure 2(b)). The results for three consecutive frames of the NYSE sequence are presented in Figure 4. Column (a) shows a sequence of the original frames; the second column, (b), shows the object delineation produced by using only pixels' intensities; The errors in these frames, mainly noticeable in the lower-left part of the images, are associated with the similarity of the color schemes of the foreground and background flags. The results using solely motion data are given in column (c). Here the error is attributed to similarity in the motion behavior: The top part of the foreground (small) flag, that is anchored, exhibits slower motion in comparison to the rest of the flag. In that it has similar motion behavior to that of the background (big) flag. The object tracking resulted by using both intensities and motion is presented in column (d) of Figure 4. This final output presents better and more accurate object tracking than the previous two. Thus it is evident that using both color and motion results in the best segmentation. This notion is substantiated by the tracking errors that appear in the same image regions both in column (b) and (c), just left to the small flag. Hence, solely color and motion can not make the separation between the flag and its background. However, when both are combined the delineation becomes more accurate.

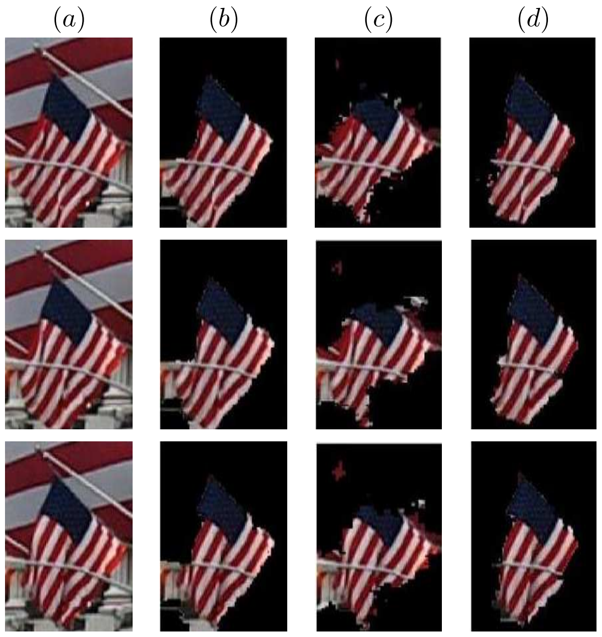

Figure 4. Flag Tracking Results - Column (a) - original frames; (b) results produced by using only pixels' intensities; (c) results using solely motion data; and (d) results by using both intensities and motion

Figures 5 presents the tracking results for two surveillance sequences taken from the CAVIAR data set.[24] These sequences present two scenarios, where the target of interest is moving (first row) and where it is standing (second row). In both cases the target of interest is occluded part of the time. The tracker position is marked with a rectangle. As can be seen in all figures, tracker sticks to the target of interest even under occlusion.

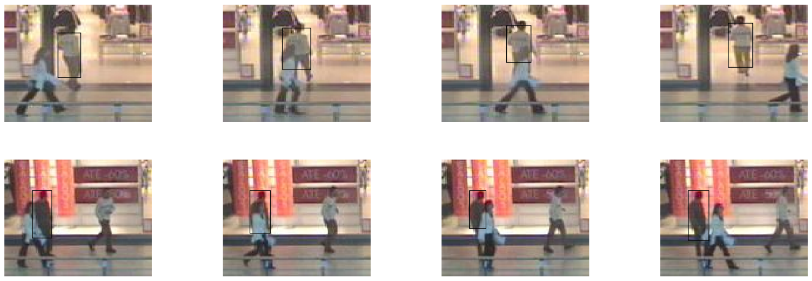

Figure 5. Surveillance Sequences Tracking Results. 4 representing frames from surveillance sequences taken from the CAVIAR data sets[24]

A synthetic sequence, taken from the standard HDTV TRICTRAC data set[30] is presented in Figure 6. Since this sequence is a synthetic one, the players' shirts' color schemes are identical. The target of interest is the red player sprinting to the right. When color data is used, both players in red are delineated, while when incorporating motion, as can be seen in Figure 6, only the player who is the target of interest is tracked.

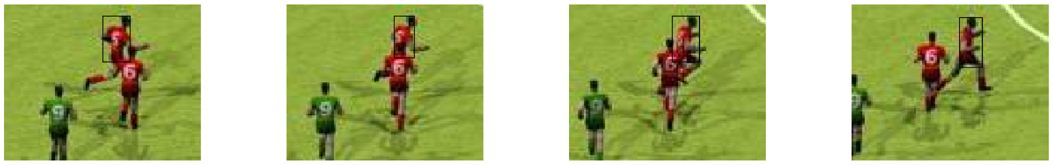

Figure 6. Tracking in Synthetic Video, of object with highly similar color scheme to other objects in the sequence

5. CONCLUSIONS

We show here a scheme for target tracking in videos that incorporates both color and motion data. The scheme presented is based on the *normalized cuts'* segmentation criterion,[13] which is solved by the HPF polynomial time algorithm.[26]

The tracking scheme presented in this paper is highly robust, thus permitting the utilization of block-matching motion estimation techniques, which are computationally efficient. The evaluation of the method on standard and non-standard benchmark videos clearly shows that the method presents comparable results to other state-of-the-art methods, while incorporating coarse and inaccurate motion field. These, along with the time efficiency of the algorithm, make our scheme a perfect choice for many online video tracking applications.

ACKNOWLEDGMENTS

The first author was partially funded by the Technion Center of Excellence in Exposure Science and Environmental Health, New-York Metropolitan and the Technions Security Science and Technology research funds, and the CITI-SENSE project of the 7th European Framework Program (FP7), ENV.2012.6.5-1.

REFERENCES

[1] Brox, T., Rousson, M., Deriche, R., and Weickert, J., "Colour, texture, and motion in level set based segmentation and tracking," *Image and Vision Computing* **28**(3), 376 – 390 (2010).

[2] Olszewska, J., De Vleeschouwer, C., and Macq, B., "Multi-feature vector flow for active contour tracking," in [*Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*], 721 –724 (31 2008-april 4 2008).

[3] Benfold, B. and Reid, I., "Stable multi-target tracking in real-time surveillance video," in [*Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*], 3457 –3464 (june 2011).

[4] Yu, J., Farin, D., and Schiele, B., "Multi-target tracking in crowded scenes," in [*Pattern Recognition*], Mester, R. and Felsberg, M., eds., *Lecture Notes in Computer Science* **6835**, 406–415, Springer Berlin / Heidelberg (2011).

[5] Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., and Gool, L. V., "Online multi-person tracking-by-detection from a single, uncalibrated camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2010). in press.

[6] Xu, N., Ahuja, N., and Bansal, R., "Object segmentation using graph cuts based active contours," *Computer Vision and Image Understanding* **107**(3), 210 – 224 (2007).

[7] Freedman, D. and Turek, M., "Illumination-invariant tracking via graph cuts," in [*Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*], **2**, 10 – 17 vol. 2 (june 2005).

[8] Malcolm, J., Rathi, Y., and Tannenbaum, A., "Multi-object tracking through clutter using graph cuts," in [*Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*], 1 –5 (oct. 2007).

[9] Bugeau, A. and Pérezz, P., "Track and cut: simultaneous tracking and segmentation of multiple objects with graph cuts," *J. Image Video Process.* **2008**, 3:1–3:14 (January 2008).

[10] Papadakis, N. and Bugeau, A., "Tracking with occlusions via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **33**, 144 –157 (jan. 2011).

[11] Lucas, B. D. and Kanade, T., "An iterative image registration technique with an application to stereo vision," in [*Intl. Joint Conf. on Artificial Intelligence (IJCAI)*], 674–679 (1981).

[12] Brox T., Bruhn A., P. N. and J., W., "High accuracy optical flow estimation based on theory for wrapping," in [*ECCV 2004*], **4**, 25–36 (2004).

[13] Hochbaum, D. S., "Polynomial time algorithms for ratio regions and a variant of normalized cut," *IEEE Trans. Pattern Analysis and Machine Intelligence* **32**(5), 889–898 (2010).

[14] Ali, S. and Shah, M., "Human action recognition in videos using kinematic features and multiple instance learning," *IEEE Trans, Pattern Analysis and Machine Intelligence* **32**(2), 288–303 (2010).

[15] Liu, J., Ali, S., and Shah, M., "Recognizing human actions using multiple features," in [*CVPR*], (2008).

[16] Zhu, S. and Ma, K.-K., "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Processing* **9**(2), 287–290 (2000).

[17] Kramer, P., Hadar, O., Benois-Pineau, J., and Domenger, J.-P., "Super-resolution mosaicing from mpeg compressed video," *Signal Processing: Image Communication* **22**(10), 845 – 865 (2007).

[18] Fishbain, B., Yaroslavsky, L., and Ideses, I., "Real-time stabilization of long range observation system turbulent video," *Journal of Real-Time Image Processing* **2**(1), 11–22 (2007).

[19] Sabirin, H., Kim, J., and Kim, M., "Graph-based object detection and tracking in h.264/avc bitstreams for surveillance video," in [*Multimedia and Expo (ICME), 2011 IEEE International Conference on*], 1 –6 (july 2011).

[20] Shi, J. and Malik, J., "Normalized cuts and image segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence* **22**(8), 888–905 (2000).

[21] Nie, Y. and Ma, K.-K., "Adaptive rood pattern search for fast block-matching motion estimation," *IEEE Trans. Image Processing* **11**(12), 1442–1449 (2002).

[22] Tourapis, A. M., Au, O. C., and Liou, M. L., "Highly efficient predictive zonal algorithms for fast block-matching motion estimation," *IEEE Trans Circuits and Systems for Video Technology* **12**(10), 934–947 (2002).

[23] VideoLAN Organization, "x264 video encoding into the h.264/mpeg-4 avc format," (2010).

[24] CAVIAR Data Set, "http://homepages.inf.ed.ac.uk/rbf/caviar," (2001).

[25] Fishbain, B., "New york stock exchange facade, video and motion," (2012).

[26] Hochbaum, D. S., “The pseudoflow algorithm: A new algorithm for the maximum-flow problem,” *Oper Res.* **56**(4), 992–1009 (2008).

[27] Chandran, B. G. and Hochbaum, D. S., “A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem,” *Oper. Res.* **57**(2), 358–376 (2009).

[28] Fishbain, B., Hochbaum, D. S., and Mueller, S., “The pseudoflow algorithm for minimum cut in vision problems.” arXiv:1007.4531v2 [cs.CV] (2009).

[29] Hochbaum, D., “HPF: Hochbasum’s pseudo-flow algorithm implementation,” (2010).

[30] TRICTRAC Test Data Sets, “http://www.multitel.be/trictrac/,” (2004).

[31] PETS 2009 Benchmark Dataset, “http://www.cvg.rdg.ac.uk/pets2009/index.html,” (2009).