Dense Spatio-Temporal Motion Segmentation for Tracking Multiple Self-Occluding People

Martin Hofmann, Gerhard Rigoll
Technische Universität München, Germany
80333 München, Germany
{martin.hofmann, rigoll}@tum.de

Thomas S. Huang
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
t-huang1@illinois.edu

Abstract

In this paper, we describe a new dense spatio-temporal motion segmentation algorithm with application to tracking of people in crowded environments. The algorithm is based on state-of-the-art motion and image segmentation algorithms. We specifically make use of a mean shift image segmentation algorithm and two graph based motion segmentation algorithms. The resulting motion segmentation is on the one hand accurate and on the other hand computationally efficient. In addition our method is capable of handling mutual occlusions. This shows that motion segmentation can efficiently be used to simultaneously detect, track and segment moving objects. We apply this to tracking people in surveillance videos, but the algorithm is not limited to this class of scenes.

1. Introduction

Motion segmentation is a method for segmenting a video sequence into segments of coherent motion. In addition to a mere segmentation into some segments of similar motion, our algorithm is capable of finding long term motion segments. Furthermore the proposed algorithm is able to associate motion segments through visual occlusion by other objects. Therefore, the found motion segments reveal individual entities that move through a scene.

The final result can therefore be called a tracking algorithm because it is able to find and track individual people in surveillance videos. It is important to note, however, that this method originates from the idea of motion segmentation and not from a tracking approach. We have shown that motion features alone give very precise tracking and segmentation at the same time, without the use of any appearance information.

We first present motivation and overview of the algorithm and then explain every component in detail. Results and a Conclusion are given at the end of the paper.

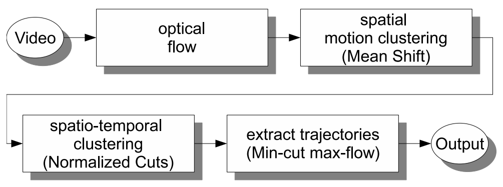
Video
optical flow
spatial motion clustering (Mean Shift)
spatio-temporal clustering (Normalized Cuts)
extract trajectories (Min-cut max-flow)
Output

Figure 1. Schematic of the dense motion segmentation method.

2. Overview and motivation

As described above, the central premise of this algorithm is to analyze the motion vector field generated from a spatio-temporal intensity pattern (a video). The goal is to find groups of points in this three-dimensional space that belong together.

Clustering and segmentation algorithms are ideally suited to find these groups of points. The disadvantage of state-of-the-art algorithms, like mean-shift or normalized cuts, is their high computational complexity and their inability to explicitly model occlusions.

Our algorithm is based on the N-cut motion segmentation algorithm [5]. Here a graph with every node representing a pixel is built and then segmented. Because this results in a huge graph, the original algorithm only works on a very subsampled version of the original video.

We overcome this problem by adding a preprocessing step: We use an image segmentation technique, namely mean shift segmentation [2], and apply it to do a pre-segmentation of the motion vector field at each frame. This greatly reduces the number of motion observations from the number of pixels in a frame to the number of motion segments in each frame.

The N-cut algorithm can find independent motions, but it is not capable of matching motion segments through occlusions. Therefore a post-processing step is added to join the clusters found by the N-cut algorithm and to overcome the problem of mutual occlusions.

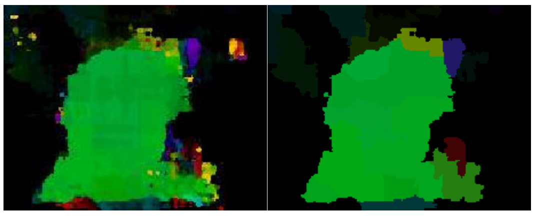

Figure 2. HSV based visualization of the optical flow (a) and the same motion field after mean shift segmentation (b).

Figure 1 shows a schematic overview of the proposed dense spatio-temporal motion segmentation algorithm. In the first step, an optical flow algorithms is applied at each pair of frames to generate the motion field. At each frame separately this motion vector field is reduced in size by mean shift segmentation. Then the N-cut algorithm is applied to find spatio-temporal clusters. In the last step, clusters are grouped together with a min-cost max-flow algorithm, which results in the final segmentation.

2.1. Definitions and optical flow calculation

The incoming video is represented as a space-time intensity pattern. Let $\mathcal{P}$ denote the set of all points in this spatio-temporal cube and let $\mathcal{P}_t \subset \mathcal{P}$ denote all points in frame t. Then obviously $\mathcal{P} = \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_N$, where N is the number of frames in the cube of interest.

Each point $s \in \mathcal{P}$ is associated with a color intensity vector $\mathbf{c}(s)$ and a motion vector $\mathbf{z}(s)$. We use the optical flow algorithm described in [4] to calculate $\mathbf{z}(s)$. This algorithm is an optical flow method based on block-matching.

2.2. Feature space reduction using mean shift

As mentioned in Section 2, the mean shift filtering is applied to the motion vector field for each frame separately.

Each frame $\mathcal{P}_t$ is segmented into a set of segments $\mathcal{S}_t^{(i)}$. This ensures that

$$\mathcal{P}_t = \mathcal{S}_t^{(1)} \cup \ldots \cup \mathcal{S}_t^{(m(t))} \cup \mathcal{B}_t \tag{1}$$

where $\mathcal{B}_t$ denotes the set of pixels that belong to the background. Background pixels are those that do not move: $\mathcal{B}_t = \{s | \ \mathbf{z}(s) = 0, \ s \in \mathcal{P}_t\}$.

The segmentation (1) is done with the mean shift algorithm. This is a straightforward application of this well known and highly optimized algorithm. The only difference is that the feature points which are being clustered are not $(2+3)$-dimensional, but only $(2+2)$-dimensional. The spatial dimension stays at two, but in the range domain we have two motion vectors instead of three color components.

The segmentation returns the mean velocity vectors $\hat{\mathbf{z}}_t^{(i)}$ for each segment, which can formally be defined as:

$$\hat{\mathbf{z}}_t^{(i)} = \frac{\sum_{s \in \mathcal{S}_t^{(i)}} \mathbf{z}(s)}{|\mathcal{S}_t^{(i)}|} \tag{2}$$

Figure 2b shows the results of the segmentation method. In this figure, the motion vector field is visualized using the mapping to the HSV color space[1]. In this image the problem size is reduced to roughly 200 clusters. In comparison to the optical flow image (Figure 2a), the differences are minor. This confirms that the clustering of the motion vectors is a valid way of reducing the problem size without losing essential information.

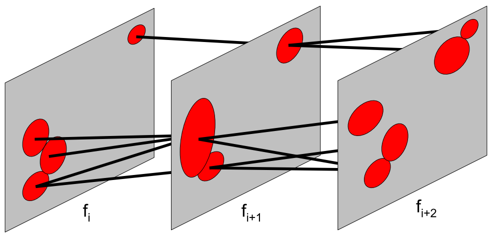

Figure 3. Associating patches to build a graph.

2.3. Carving blocks using normalized cuts

The N-cut algorithm [5] clusters data based on an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The graph is represented by its weight matrix W. In order to apply N-cut segmentation, this weight matrix has to be built first.

In the mean shift segmentation step, a set of motion coherent segments has been generated at each frame. Each of these segments for all frames is associated with a node in the graph $\mathcal{G}$. Figure 3 illustrates this. The N-cut algorithm does not require all nodes to be connected. In fact, we only connect segments which overlap with segments in Δt of the adjacent frames.

The affinity between the adjacent elements should have two properties: (1) Affinity should be high if the segments are likely to belong together spatially and (2) the affinity should be high if the segments have similar motion. Each segment is therefore projected to its adjacent frames using the mean motion vector. Then the overlap with segments present in that frame is calculated, as well as their motion similarity.

More precisely, for each frame t, each segment $\mathcal{S}_t^{(i)}$ is projected to its $\Delta t = 1 \ldots \Delta t_{\max}$ adjacent frames using the mean motion vector:

$$\mathcal{S}_{t+\Delta t|t}^{(i)} = \left\{ s + \Delta t \cdot \hat{\mathbf{z}}_t^{(i)} | \ s \in \mathcal{S}_t^{(i)} \right\} \tag{3}$$

[1] The direction of the motion vector is mapped to the H-component, the V-component is set to the magnitude and the S-component is set to one.

The cut set of the projection with the points in the corresponding frame is given by

$$\hat{\mathcal{S}}^{(i)}_{t+\Delta t|t} = \mathcal{S}^{(i)}_{t+\Delta t|t} \cap \mathcal{P}_{t+\Delta t} \qquad (4)$$

This cut set reveals the overlap of the i-th segment in frame t with all the segments j in frame $t+\Delta t$. More precisely, the overlapping area of the j-th segment in frame $t+\Delta t$ with the projected segment i of frame t is given by

$$A^{(i\to j)}_{t\to t+\Delta t} = \left|\mathcal{S}^{(j)}_{t+\Delta t} \cap \hat{\mathcal{S}}^{(i)}_{t+\Delta t|t}\right| \qquad (5)$$

The motion similarity is measured by

$$d^{(i\to j)}_{t\to t+\Delta t} = \left\|\hat{\mathbf{z}}^{(j)}_{t+\Delta t} - \hat{\mathbf{z}}^{(i)}_{t}\right\| \qquad (6)$$

The affinity between segment i in frame t and segment j in frame $t+\Delta t$ is then defined as

$$a^{(i\to j)}_{t\to t+\Delta t} = A^{(i\to j)}_{t\to t+\Delta t} \cdot \exp(-(d^{(i\to j)}_{t\to t+\Delta t})^2/2\sigma^2) \qquad (7)$$

All the affinity measures a are plugged into the affinity matrix W, thus forming the undirected graph G. Since the affinities are only processed in one direction, the graph affinity matrix will have upper triangular form. To build an undirected graph, the symmetric affinity matrix is calculated using $\hat{W} = W + W^T$.

This graph can now be segmented using the N-cut algorithm. The algorithm cuts the graph into two pieces and repeats this procedure recursively. The clustering is stopped when the N-cut value drops below a predefined threshold which is set so as to oversegment the scene into relevant chunks. This threshold is a function of the definition of the affinity value a, and of the size of the chunks that one wants to find. The size of the chunks is known for a given class of videos and thus has to be set only once.

After the segmentation process, the scene $\mathcal{P}$ is clustered into disjoint spatio-temporal blocks $\mathcal{C}_i$:

$$\mathcal{P} = \mathcal{C}_{(1)} \cup \ldots \cup \mathcal{C}_{(i_{\max})} \cup \mathcal{B} \qquad (8)$$

where $\mathcal{B}$ denotes all the spatio-temporal points that belong to neither object and thus are background.

The block's centroid $\hat{\mathbf{c}}_i$ and the mean velocity vector $\hat{\mathbf{z}}_i$ are defined as

$$\hat{\mathbf{c}}_i = \frac{\sum_{s\in\mathcal{C}_i} s}{|\mathcal{C}_i|} \qquad (9)$$

$$\hat{\mathbf{z}}_i = \frac{\sum_{s\in\mathcal{C}_i} \mathbf{z}(s)}{|\mathcal{C}_i|} \qquad (10)$$

Figure 4 shows the results of the normalized cut clustering procedure on the surveillance video. Interesting to note is that the big chunks of motion have been found successfully. Those chunks do not necessarily have to have all the same motion; the motion only has to be similar from one frame to the next. Therefore, non-linear motions can also be correctly clustered into single blocks. Note also that the scene is still widely oversegmented and complete trajectories have not been found yet. The reason for this oversegmentation is given in the next section.

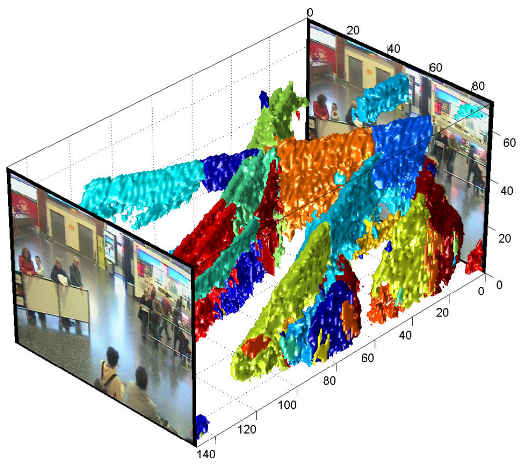

Figure 4. Results of the spatio-temporal normalized cut clustering.

3. Global data association to find trajectories

In the original normalized cut segmentation algorithm, the clusters resulting from the N-cut step are declared to be the final segmentation. The problem with this method is the inability to handle occlusions. The trajectory of a moving object which is divided in the middle by another occluding object will be clustered into two segments. This is because there will be no affinities between the two ends of the two pieces.

In the previous section we have explained how affinities are calculated between segments that are up to Δt frames apart. This readily allows us to recover short time occlusions. One could just increase Δt in order to increase the ability to handle bigger occlusions, but that would lead to an exponential growth of the computational complexity. Instead, we propose another approach: We intentionally perform the segmentation in the normalized cut clustering so as to oversegment the scene. This way, final trajectories are not found yet, but the identified blocks $\mathcal{C}_i$ are likely to belong to single coherent blocks of motion.

Knowing single coherent pieces allows us to reason about occlusion. In the final processing step, we use a data association technique to stitch these pieces together into trajectories in a way to account for longterm occlusions. A MAP definition is used of matching observations to trajectories. This description can be mapped into a min-cost max-flow network, which then can be solved with linear programming methods.

3.1. Finding Trajectories with a MAP Approach

Let $\mathcal{X} = \{\mathbf{x}_i\}$ be the set of observations. In our case the observations are $\mathbf{x}_i = \{\mathcal{C}_i,\ \hat{\mathbf{c}}_i,\ \hat{\mathbf{z}}_i\}$, where $\mathcal{C}_i$ is the set of points belonging to block i, $\hat{\mathbf{c}}_i$ is the centroid of the block and $\hat{\mathbf{c}}_i$ is the mean velocity vector over all points of the block.

A trajectory is defined as the set of observations that belong to the trajectory: $\mathcal{T}_k = \{\mathbf{x}_{k_1}, \mathbf{x}_{k_1}, \ldots, \mathbf{x}_{k_l}\}$. More than one trajectory is possible, so that the whole set of observations is given by $\mathcal{T} = \{\mathcal{T}_k\}$. We want to find the optimal set of trajectories given the observation. The MAP formulation is given by

$$\begin{aligned}\mathcal{T}^* &= \arg\max_{\mathcal{T}} P(\mathcal{T}|\mathcal{X}) \\ &= \arg\max_{\mathcal{T}} P(\mathcal{X}|\mathcal{T})P(\mathcal{T}) \\ &= \arg\max_{\mathcal{T}} \prod_i P(x_i|\mathcal{T})P(\mathcal{T}) \qquad (11)\end{aligned}$$

The last step in Equation (11) assumes conditional independence of the likelihood probabilities given $\mathcal{T}$. Optimizing this equation directly is infeasible due to the huge number of possible combinations of trajectories. Instead a non-overlap constraint is introduced that states that a given observation can only belong to one trajectory.

$$\mathcal{T}_k \cap \mathcal{T}_l = \emptyset,\ \forall k \neq l \qquad (12)$$

Furthermore, it is assumed that the trajectories $\mathcal{T}_k$ are independent of each other. Then using the constraint (12), Equation (11) can be written as:

$$\begin{aligned}\mathcal{T}^* &= \arg\max_{\mathcal{T}} \prod_i P(x_i|\mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k) \qquad (13) \\ & \text{s.t. } \mathcal{T}_k \cap \mathcal{T}_l = \emptyset,\ \forall k \neq l \qquad (14)\end{aligned}$$

The first term of Equation (13) is the likelihood function of observations. A Bernoulli distribution is used to model the observation probability. Here β_i denotes the false alarm probability of observing the object.

$$P(x_i|\mathcal{T}) = \begin{cases} 1-\beta_i & \exists \mathcal{T}_k \in \mathcal{T}, x_i \in \mathcal{T}_k \\ \beta_i & \text{otherwise} \end{cases} \qquad (15)$$

The second term of Equation (13) is modeled as a Markov chain with initialization probability P_{entr}, termination probability P_{exit} and transition probability P_{link}:

$$\begin{aligned}P(\mathcal{T}_k) &= P(\{x_{k_0}, x_{k_1}, \ldots, x_{k_{l_k}}\}) \\ &= P_{entr}(x_{k_0})P_{link}(x_{k_1}|x_{k_0})\ldots \\ &\quad P_{link}(x_{k_{l_k}}|x_{k_{l_k-1}})P_{exit}(x_{k_0}) \qquad (16)\end{aligned}$$

The definition of these probabilities and their relation to the observed clusters is described later in Section 3.3.

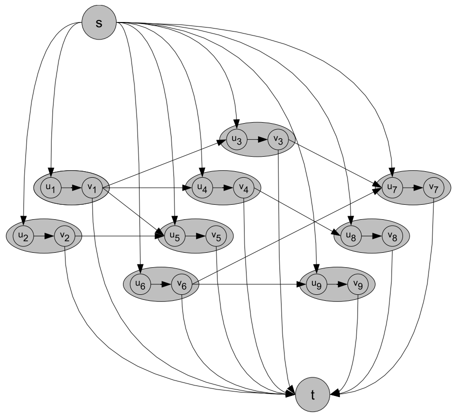

Figure 5. The cost flow network.

3.2. Mapping to a Min-Cost Max-Flow Network

The basic idea is that non-overlaping trajectories can be mapped to a cost flow graph, as has been proposed in [8].

In this work, a cost flow graph is used for the data association step. Figure 5 illustrates such a graph. In this, each observation is represented by two nodes connected with a directed edge. This edge holds the cost of using this observation. Each node pair is connected to observations nearby, also with a directed edge. This edge holds the cost of associating the connected node pairs. Furthermore, each node pair is connected to a global source and the sink node. In addition to the edge costs, each edge is set to have at most one unit of flow. This allows for modeling of the non-overlap constraint. A flow, starting at the source node, searches its way through the graph to reach the sink node in a way that minimizes the overall cost.

A mathematical definition of the flow network will be given next. To this end, flow indicator functions holding only 0 or 1 are defined:

$$f_{en,i} = \begin{cases} 1 & \exists \mathcal{T}_k \in \mathcal{T}, \mathcal{T}_k \text{ starts from } x_i \\ 0 & otherwise \end{cases} \qquad (17)$$

$$f_{ex,i} = \begin{cases} 1 & \exists \mathcal{T}_k \in \mathcal{T}, \mathcal{T}_k \text{ ends from } x_i \\ 0 & otherwise \end{cases} \qquad (18)$$

$$f_{i,j} = \begin{cases} 1 & \exists \mathcal{T}_k \in \mathcal{T}, x_j \text{ is after } x_i \text{ in } \mathcal{T}_k \\ 0 & otherwise \end{cases} \qquad (19)$$

$$f_i = \begin{cases} 1 & \exists \mathcal{T}_k \in \mathcal{T}, x_i \in \mathcal{T}_k \\ 0 & otherwise \end{cases} \qquad (20)$$

For a given set of trajectories $\mathcal{T}$ these indicator functions are determined. Therefore any valid configuration of the indicator functions also determines the trajectories. $\mathcal{T}$ is non-overlap if and only if

$$f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j},\ \forall i \tag{21}$$

It can be shown [8], that the maximum a posteriori formulation of Equation (13) is equivalent to calculating the max flow, if and only if the costs associated with each graph edge are defined as

$$\begin{aligned} C_{en,i} &= -\log P_{entr}(x_i) & (22)\\ C_{ex,i} &= -\log P_{exit}(x_i) & (23)\\ C_{i,j} &= -\log P_{link}(x_j|x_i) & (24)\\ C_i &= \log \frac{\beta_i}{1-\beta_i} & (25)\end{aligned}$$

Note that all the costs are positive except for C_i, which may be negative if $\beta_i < 0.5$.

For each observation $\mathbf{x}_i$ two nodes u_i, v_i are created. In addition, one source node s and one sink node t are created.

The costs and flows are set as follows: $c(u_i, v_i) = C_i$, $f(u_i, v_i) = f_i$; $c(s, u_i) = C_{en,i}$, $f(s, u_i) = f_{en,i}$; $c(v_i, t) = C_{ex,i}$, $f(v_i, t) = f_{ex,i}$; $c(v_i, u_j) = C_{i,j}$, $f(v_i, u_j) = f_{i,j}$.

The max-flow problem can be solved with the Ford-Fulkerson algorithm, Karger's algorithm [1], or the scaling push-relabel method [3]. We use the Torsche scheduling toolbox [7], which uses linear programming to solve the flow problem.

The optimal flows returned by the algorithm are mapped back to find the trajectories using the indicator functions (17)-(20). The number of trajectories found by the optimization is dependent on the flow sent from node s to node t. If the flow is set to q, then q non-overlapping trajectories are found. Therefore for different flows, different solutions are obtained. We simply run the optimization algorithms for all possible values of q and take the q which minimizes the flow costs.

$$q^* = \arg\max_q f(\mathcal{G}(q)) \tag{26}$$

where $f(\mathcal{G}(q))$ is the flow returned by optimizing the graph $\mathcal{G}$ set with flow q. It can be shown that instead of brute-force enumeration of all possible q values, a binary search can be applied, because $f(\mathcal{G})$ is a convex funtion in q.

3.3. Estimating Model Parameters from Clustering

For the algorithm to work as desired it is essential to have good estimates for the model probabilities, namely P_{entr}, P_{exit}, P_{link} as well as for the false alarm rate β.

In our case, the observations are defined as $\mathbf{x}_i = \{\mathcal{C}_i,\ \hat{\mathbf{c}}_i,\ \hat{\mathbf{z}}_i\}$, where $\mathcal{C}_i$ it the set of points belonging to the block, $\hat{\mathbf{c}}_i$ is the centroid of the block and $\hat{\mathbf{c}}_i$ is the mean velocity vector over all points of the block.

We observe that trajectories are likely to be true observations if they are big. We therefore model the false alarm rate as:

$$\beta_i = \left(\frac{1}{2}\right)^{\frac{|\mathcal{C}_i|}{\mu_\beta}} \tag{27}$$

where $|\mathcal{C}_i|$ denotes the volume of cluster i and μ_β denotes the reference volume. Any object bigger than the reference volume will result in a $\beta_i < 0.5$ and thus will have negative cost in the flow graph. Those objects therefore are encouraged to be part of some trajectory. Consequently, objects with volume smaller than μ_β will have positive cost and are only added in the graph if they help to form a trajectory.

To calculate the transition probabilities $P_{link}(\mathbf{x}_j|\mathbf{x}_i)$, an affinity measure has to be generated. This step is crucial in overcoming the occlusion problem. Clusters are likely to belong together if they have (1) high similarity in their mean velocity and (2) high spatial continuity.

To meet the first requirement, the motion similarity is measured by

$$d_{j|i} = \exp(-\left\|\hat{\mathbf{z}}_i - \hat{\mathbf{z}}_j\right\|^2 / 2\sigma^2) \tag{28}$$

For the second requirement, the spatial continuity is measured. The key to do this is the use of 3D morphologic operations. We use a morphologic dilate operation for each observation separately using a structure element which encodes the motion of the object. Doing so extends the 3D spatio-temporal blob of the object in time.

First, each object i is dilated, which results in a spatio-temporal prediction:

$$\hat{\mathcal{C}}_i = (\mathcal{C}_i \oplus \mathcal{M}_i) \setminus \mathcal{C}_i \tag{29}$$

where the $\oplus$-operator denotes morphologic dilate and $\mathcal{M}_i$ is a structure element which has the form of a line from the origin in the direction of the mean velocity vector $\Delta t \cdot \hat{\mathbf{z}}_i$. In our experiments $\mathcal{M}_i$ is generated with the Bresenham line interpolation algorithm. The scaling factor Δt specifies the length of the structure element and thus defines the number of frames the cluster is projected into the future.

The spatial overlap is calculated as

$$s_{j|i} = \left|\hat{\mathcal{C}}_i \cap \mathcal{C}_j\right| \tag{30}$$

The link probability is then assembled to take spatial and motion continuity into account:

$$P_{link}(\mathbf{x}_j|\mathbf{x}_i) = \frac{s_{j|i} \cdot d_{j|i}}{\sum_j s_{j|i} \cdot d_{j|i}} \tag{31}$$

The enter and exit probabilities are modeled explicitly. In our video sequences, objects are likely to appear on either the left or the right side of the frame, or the lower edge of the frame. In addition, objects might already exist in

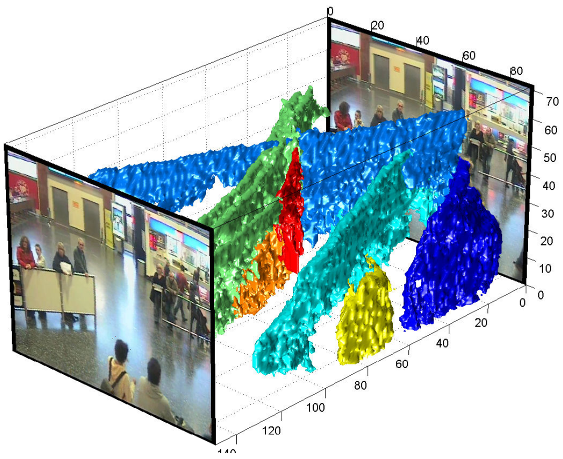

Figure 6. Results of the dense motion segmentation. This figure shows the spatio-temporal blocks that have been automatically segmented.

the first frame. Therefore any object that exists in the first frame is likely to be the beginning of a trajectory. Analogously, object trajectories must be terminated at the end of the sequence and therefore have high termination probability near the end. The probabilities are therefore modeled appropriately in a way to capture these relations.

4. Results

Results are shown for a video sequence from the Trecvid dataset [6]. The final clustering for this surveillance video is seen in Figure 6, which shows a 3D visualization of the found clusters in the spatio-temporal data volume. Figure 7 depicts a selected frame from this sequence. In this frame, the sequence was segmented into six clusters.

These results show that the major motions are automatically found and correctly labeled. Small cluttered blocks which would result from noise have successfully been removed. Important to note is that the method is capable of keeping identity through occlusions.

It is also very interesting to note that this kind of segmentation not only finds major motions, but is also able to very precisely segment the objects. This is remarkable, given that only motion information and no appearance information is processed.

However, there are also some false positives. The orange cluster, for example, would be considered such a false positive. This cluster somewhat detects the shadow and the feet of the person labeled in green. Ideally this green person and the orange feet cluster should be clustered together. The reason for this false positive is that the shadow seems to be a significant independent motion. This shows that the algorithm is not mainly a people tracking algorithm, but rather a generic motion segmentation algorithm. In order to gear the algorithm more to a specific domain like people tracking,

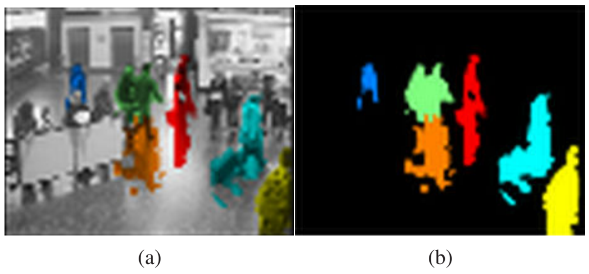

Figure 7. (a) Original frame from a surveillance video; (b) Segmentation results.

prior knowledge about the object class would be necessary.

4.1. Conclusion

In this paper we have presented a motion segmentation algorithm which extends the normalized cut motion segmentation method and allows to (1) handle long scenes (2) at high resultion and (3) is capable of handling a significant amout of mutual occlusions. It can be seen that motion segmentation alone can simultaneously give detection, tracking and segmentation. We have shown that this method can readily be applied to tracking multiple highly articulated objects like humans.

References

[1] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[2] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *Proc. Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1197–1203, Sept. 1999.

[3] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22(1):1–29, 1997.

[4] A. S. Ogale and Y. Aloimonos. A roadmap to the integration of early visual modules. *International Journal of Computer Vision*, 72(1):9–25, 2007.

[5] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *Proc. Sixth International Conference on Computer Vision*, pages 1154–1160, Jan. 1998.

[6] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and TRECVid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, 2006.

[7] P. Šůcha, M. Kutil, M. Sojka, and Z. Hanzálek. TORSCHE scheduling toolbox for MATLAB. In *IEEE Computer Aided Control Systems Design Symposium (CACSD'06)*, pages 1181–1186, Munich, Germany, October 2006.

[8] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, pages 1–8, June 2008.