Clustering Motion for Real-Time Optical Flow based Tracking

Tobias Senst, Rubén Heras Evangelio, Ivo Keller, Thomas Sikora
Communication Systems Group
Technische Universität Berlin
Berlin, Germany
senst,heras,keller,sikora@nue.tu-berlin.de

***Abstract*—The selection of regions or sets of points to track is a key task in motion-based video analysis, which has significant performance effects in terms of accuracy and computational efficiency. Computational efficiency is an unavoidable requirement in video surveillance applications. Well established methods, e.g. Good Features to Track, select points to be tracked based on appearance features such as cornerness and therefore neglecting the motion exhibited by the selected points. In this paper, we propose an interest point selection method that takes into account the motion of previously tracked points in order to constrain the number of point trajectories needed. By defining pair-wise temporal affinities between trajectories and representing them in a minimum spanning tree, we achieve a very efficient clustering. The number of trajectories assigned to each motion cluster is adapted by initializing and removing tracked points by means of feed-back. Compared to the KLT tracker, we save up to 65% of the points to track, therefore gaining in efficiency while not scarifying accuracy.**

***Keywords*-feature tracking; long-term trajectories; optical flow; RLOF**

I. Introduction

Analyzing motion and moving objects in video data provides relevant information for computer vision and surveillance systems. Optical flow is increasingly being used in video surveillance systems, e.g., in order to distinguish people by motion in crowds, for crowd analysis and activity monitoring.

A common motion representation is provided by point trajectories, which require a fast and accurate motion tracking for large sets of points. Most recent work [1], [2] has been motivated by the huge improvements in dense optical flow estimation [3]. A method dealing with dense sets of trajectories is e.g. the Particle Video [4], which is based on extracting point trajectories from a dense optical flow method, but it is rather slow. Furthermore, parallel computing has emerged as a key technique in order to increase the performance of motion computation. Fast and parallelized implementations [5], [6] enable the computation of dense motion fields using the GPU. In [7] a large displacement optical flow (LDOF) as introduced in [6] is used to create dense point trajectories with a high performance. Nevertheless, with a run-time of above 1000 ms, it is still not suitable for surveillance applications.

Due to its high computational efficiency the Kanade Lucas Tomasi feature tracker (KLT) [8] still remains as a widely accepted method utilized to compute sparse motion fields or trajectories in video sequences [9], [10]. While most global optical flow techniques, whose computational effort is tied to the image size, outperform the KLT method in terms of accuracy and density of the computed motion fields, the KLT is faster for sparse motion fields due to the scalability of the computational complexity regarding to the number of points to be tracked. The KLT tracker employs the Good Features to Track (GFT) method in order to select a set of points to track and estimates their motion using the Lucas Kanade method. Very fast implementations of the KLT tracker [8] use parallelization [11], [5]. In [12] a robust variation of the Lucas Kanade method is presented which is able to track up to 10.000 points in real-time (>25fps).

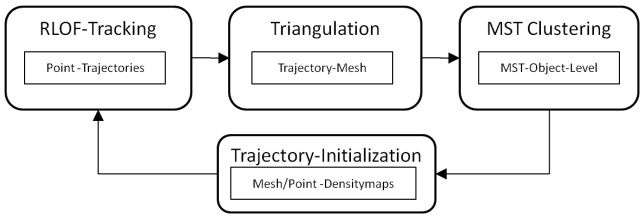

Figure 1. Process stages of the proposed approach. Motion clustering is performed at the MST representation of the trajectory graph. A feed-back loop enables to distribute tracks homogeneously to each moving objects.

The choice of the selected points to track is crucial for the performance by using the KLT tracker. State-of-the-art KLT trackers [8], [13] are still based on the GFT method as well selecting points on image regions with high cornerness [14]. Cornerness is an important selection criterion in order to avoid singularity problems of the Lucas Kanade method regarding the aperture problem. But, as stated in [12], corner regions are likely to lie on motion boundaries, thus violating the local constant motion assumption. Moreover, the extracted motion information should properly represent the whole scene, i.e. each moving object should get assigned enough trajectories to accurately describe its motion while preserving large objects, e.g., the background, from containing too many trajectories in order to avoid redundant information and thus saving computational resources. This is still not taken into account by the KLT tracker and in particular by the GFT method.

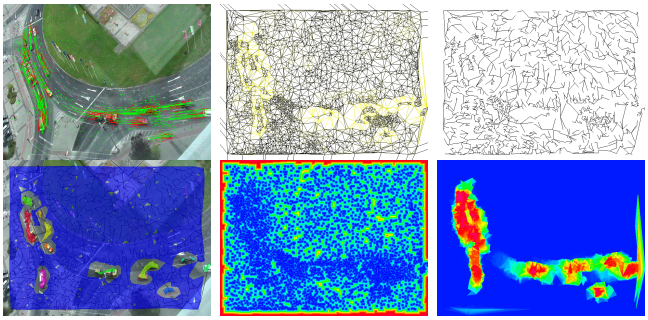

Figure 2. **From top left to lower right: (a)** Point trajectories, **(b)** triangulated point trajectories, **(c)** MST, **(d)** motion segments, **e)** point density and **(f)** mesh density map.

In this paper, we propose a method for selecting points to track by adding the constraint that each moving object should be represented by a given number of point trajectories, depending on the application characteristics. Therefore, we use a feed-back loop to adapt the number of trajectories by initializing and removing them according to specific properties of the motion segments. Fig. 1 and Fig. 2 provide an overview and a graphical representation of the proposed system, respectively. For the point tracking (Fig. 2(a)) task we chose the robust local optical flow (RLOF) as proposed in [12] due to its good accuracy and run-time relation. Motion segmentation is performed by trajectory pair-wise affinity clustering. Spatial affinity is modeled by the triangulation of the trajectory endpoints. In this way, temporal affinities are only computed for adjacent trajectory endpoints. This leads to an undirected weighted graph (Fig. 2(b)). Trajectory clustering is performed by using the minimum spanning tree (MST) of this undirected graph (Fig. 2(c)). By assuming that each motion cluster (Fig. 2(d)) represents a different moving object, a point density map (e) and a mesh density map (f) containing the trajectory density, will be computed for each motion segment. This process is thoroughly described in Section 2. In combination with the current point density, we initialize new trajectories at regions with corner like textures and low point and mesh densities, in order to obtain sparse point trajectories for big objects as the background and dense point trajectories for small objects, as we describe in Section 3. In Section 4 we present some experimental results and compare our system to state-of-the-art optical flow based trackers showing significant run-time improvements at equivalent tracking accuracy. Section 5 concludes the paper.

II. Trajectory Affinities and Clustering

Given a set of asynchronous trajectories T i.e., with different temporal elongations, for each frame t, we need a locally adaptive interconnecting mechanism for their corresponding trajectory points to separate the different motion entities. We found the Delaunay triangulation [15] to be a powerful and fast method. An undirected graph $G(E, V)$ is built from the set of trajectory endpoints $\dot{\mathbf{x}}_0, \dots \dot{\mathbf{x}}_j \in \dot{T}$ representing the set of nodes V. We assign low weights w to pairs of point trajectories that are linked through the set of edges E and have similar motion. Since the discriminability among tracks is increased with their length, only trajectories $\dot{T} \subset T$ tracked for a minimum number of frames τ_1 are selected to build the graph. The triangulation indirectly models spatial affinities and speeds-up computations but still enabling to links of not directly connected tracks through a path in the graph. Each edge is assigned to a weight representing the local temporal dissimilarity between the corresponding tracks. We choose a metric that compares the maximal distance between two tracks. For T_A and T_B:

$$d(T_A, T_B) = \underset{T_A \bigcap T_B}{max} ||T_A - T_B|| \tag{1}$$

for the overlap between trajectory T_A and T_B. We use a fixed standard exponential to turn the distances into weights.

$$w(T_A, T_B) = 1 - e^{-(0.2 \cdot d(T_A, T_B))^2} \tag{2}$$

The more A and B belong to the same moving object the more likely $d(T_A, T_B)$ and the weight w tends to be small.

The motion clustering at the object level is performed by separating the undirected weighted graph G of all valid trajectories $\dot{T}$ into connected sub-graphs $\bar{G} \subset G$, each of which represents an individual moving object. This is performed in an efficient way on the minimum spanning tree (MST) [16] of the graph G. The tree in G is a connected graph without circuits and a spanning tree of G is a tree which contains all nodes of G. From a graph G it is possible to build a set (forest) of spanning trees. The spanning tree with the minimal sum of weights is called MST. The MST is a configuration that satisfies the minimum principle [17]. Due to this principle, two clusters are linked up by a single edge of the MST. Such edges are called inconsistent. In other words, clustering is performed by finding inconsistent edges of the MST and building sub-trees by removing them.

By following the depth neighborhood criterion as introduced in [17], we find inconsistent edges e_i as those edges, whose weights w_i are larger than the average weight of nearby edges in the tree. Therefore, we consider for each pair of nodes v_s and v_t, being v_s the source and v_t the target of a given edge e_i, a depth neighborhood of depth δ. The depth neighborhood for each node is the set of edges that belong to the path of depth δ with origin at that given node, excluding the path through e_i. Let μ_s and σ_s, respectively, be the average weight and variance of the depth neighborhood of v_s. Similarly, let μ_t and σ_t be the average weight and variance of the v_t depth neighborhood. An edge e_i is considered to be inconsistent if one of the following

conditions holds:

$$w_i > \mu_s + c \cdot \sigma_s \quad (3)$$
$$w_i > \mu_t + c \cdot \sigma_t \quad (4)$$
$$w_i > max(\mu_s + c \cdot \sigma_s, \mu_t + c \cdot \sigma_t) \quad (5)$$
$$f < \frac{w_i}{max(\mu_s + c \cdot \sigma_s, \mu_t + c \cdot \sigma_t)} \quad (6)$$

,where c and f are constants. Inconsistent edges are used to disjoint the tree into sub-trees.

III. Point Trajectory Reselection

The main contribution of this paper is the improvement achieved by the initialization of trajectories. While current techniques are still based on the cornerness of the images, i.e., the texture appearance, in addition our approach is take into account clustered motion information. Therefore, we set the following conditions to each set of trajectories:

1) The number of trajectories for one motion cluster should be limited.
2) Individual clusters should be constituted by a minimal number of long-term trajectories.
3) The set of point trajectories corresponding to a cluster should be uniformly distributed.
4) The trajectories should be initialized at corner-like regions.

The probability $p_T(\mathbf{x})$ of initializing a new trajectory is composed by two independent probabilities, namely, the point density probability $p_{\dot{T}}$ and the mesh density probability $p_{\bar{G}}$.

$$p_T(\mathbf{x}) = p_{\dot{T}}(\mathbf{x}) \cdot p_{\bar{G}}(\mathbf{x}) \quad (7)$$

$p_{\dot{T}}(\mathbf{x})$ refers to the spatial distribution of the trajectories, which should be proportional to the minimal distance between x and each trajectory (as imposed by condition 3):

$$d_{\dot{T}}(\mathbf{x}) = \min_{\dot{\mathbf{x}}_j \in \dot{T}} |\mathbf{x} - \dot{\mathbf{x}}_j|, \quad (8)$$

where $d_{\dot{T}}(\mathbf{x})$ denotes the point density map. Having a huge number of trajectories, $d_{\dot{T}}(\mathbf{x})$ can be efficiently computed by using the distance transform of the $\dot{\mathbf{x}}_j$ point map. We use the exponential function and a constant σ_d to turn the distances into the point density probability:

$$p_{\dot{T}}(\mathbf{x}) = 1 - e^{-\left(\frac{d_{\dot{T}}(\mathbf{x})}{\sigma_d}\right)^2}. \quad (9)$$

To satisfy conditions (1) and (2) we integrate the motion segmentation results, see Section II. The number of nodes of the clustered sub-graph set $\bar{G} \subset G$ are projected by their respective faces onto the mesh density map $d_{\bar{G}}(\mathbf{x})$. The map $d_{\bar{G}}(\mathbf{x})$ is turned into a probability, setting the mesh density constant σ_n by:

$$p_{\bar{G}}(\mathbf{x}) = e^{-\left(\frac{d_{\bar{G}}(\mathbf{x})}{\sigma_n}\right)^2}. \quad (10)$$

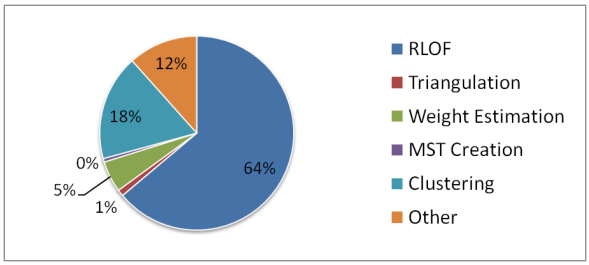

Figure 3. Partition of execution time. Trajectory affinity is measured with $\tau_1 = 10$ and trajectory could be removed with $\tau_2 = 21$ and the clustering is performed with $c = 4$, $f = 0.2$ and $\delta = 3$. To measure the mesh and point density we apply $\sigma_d = 10$ and $\sigma_n = 16$.

We use the FAST feature detector to sample the set of new trajectory candidates. The FAST [18] method is a runtime efficient corner detector. Corners are assumed at positions with not self-similar patches. This is measured by considering a circle, where the intensity from the center of the circle is compared with intensity values of each end of a line across the circle diameter. Thus, a patch is not self similar if pixels at the circle look different from the center x. A new trajectory is initialized at x conditioned by $p_T(x)$ as in Eq.(7).

Under certain conditions, especially with static cameras, trajectories could have a rather long live. Thus, we have to remove trajectories to satisfy condition one. Therefore, we define the probability $p_{T^{-1}}(\mathbf{x}) = 1 - p_{\bar{G}}(\mathbf{x})$ of removing trajectories, which have a minimal size of τ_2. The process of initializing and rejecting trajectories enables our method to maintain a nearly constant number of features for each object depending on its size.

IV. Evaluation

The proposed system has been implemented on a platform consisting of an AMD Phenom II X4 960 running at 2.99 GHz and a Nvidia GTX 275 graphic device. The RLOF is implemented on the CPU and on the GPU. The motion segmentation and point selection algorithms are only implemented on the CPU. Figure 3 shows the breakdown of the proposed method. Most of the run-time (64%) is spent on computing the point trajectories, 24% is spent on the motion segmentation and 12% is used for other processes including e.g. trajectory reselection.

A. Tracking

For the evaluation we use the MIT dataset introduced in [19], which provides the ground truth optical flow for sequences up to 75 images. We compare the proposed motion oriented robust local optical flow tracker (MORLOF) with the KLT (we used the CPU implementation provided by the OpenCV 2.0 library), RLOF and LDOF [7] tracker. Since RLOF does not include a feature selection method we use the GFT method to initialize the RLOF tracker. The

	camera (37 frames)			fish (75 frames)			hand (48 frames)		
	AED	Points tracked	Runtime per frame (ms)	AED	Points tracked	Runtime per frame (ms)	AED	Points tracked	Runtime per frame (ms)
LDOF	1.41	101662	64000	3.39	75907	320000	2.14	151018	146000
KLT	4.04	4040	1691	11.03	5201	2626	4.75	3496	1325
RLOF*	3.46	3952	1884	8.21	4823	3131	4.69	3531	1730
RLOF**	2.68	1335	353	8.06	846	157	10.01	87	77
MORLOF	2.60	1413	729	4.33	792	507	6.73	84	115

Table I
TRACKING ACCURACY OF THE LDOF, KLT, RLOF*, RLOF** AND MORLOF FOR THE MIT SEQUENCE. THE RLOF* AND RLOF** POINTS TO TRACK ARE DETECTED BY GFT WITH VARYING PARAMETERS. TIME MEASURES ARE PERFORMED ON THE CPU. (LDOF IS AVAILABLE AT HTTP://LMB.INFORMATIK.UNI-FREIBURG.DE/PEOPLE/BROX/ AND RLOF IS ABAILABLE AT HTTP://WWW.NUE.TU-BERLIN.DE/MENUE/FORSCHUNG/PROJEKTE/RLOF/.)

RLOF tracker are evaluated at two operation points by suppressing features with less texture information to illustrate the influence of different numbers of features for GFT based feature tracker. RLOF* defines the operation point with a adequate number of features, that is approximately the same as selected by the default KLT tracker. And RLOF** defines the operation point with a very low number of features, that is approximately the same as these extracted by the MORLOF. This feature set contains only features with very strong edges. Ground truth trajectories are obtained from the ground truth optical flow. The tracking error is measured as the average Euclidean distance (AED) between the trajectory endpoints and the predicted positions according to the ground truth as introduced in [7]. LDOF measurements are taken from [7] and the false tracking detections of the KLT and RLOF trackers are done regarding [12] with $\epsilon_d = 1$. The KLT has been configured to use $\Omega = 17 \times 17$ and $SSD = 100000$ in order to avoid pre-rejection of tracked points. The RLOF regions are set to $\Omega_{small} = 7 \times 7$ and $\Omega_{large} = 17 \times 17$. The MORLOF is performed with the mesh and point density variance $\sigma_d = 10$ and $\sigma_n = 16$. The trajectory clustering is performed by $c = 4$, $f = 0.2$ concerning a neighbor depth $\delta = 3$ for trajectories with $\tau_1 = 10$ and the trajectory removal with constant $\tau_2 = 21$. The evaluation has been performed with the three MIT sequences containing more than 21 frames. To measure the computational cost of the tracker we provide the number of tracked points and the run-time at the last frame of each sequence.

As Table I shows, the LDOF is outperforming the local optical flow trackers in terms of accuracy. Nevertheless, the CPU implementation of the LDOF is not practicable for real-time applications. The run-time of the CPU MORLOF implementation is clearly enhanced compared to the run-time of the GFT-based trackers (KLT and RLOF*), with an improved overall accuracy excluding the hand sequence. Obviously, the AED is not necessarily related to the number of points to track. With approximately the same number of tracked points the MORLOF the accuracy could be improved related to the RLOF**. The feature distributions

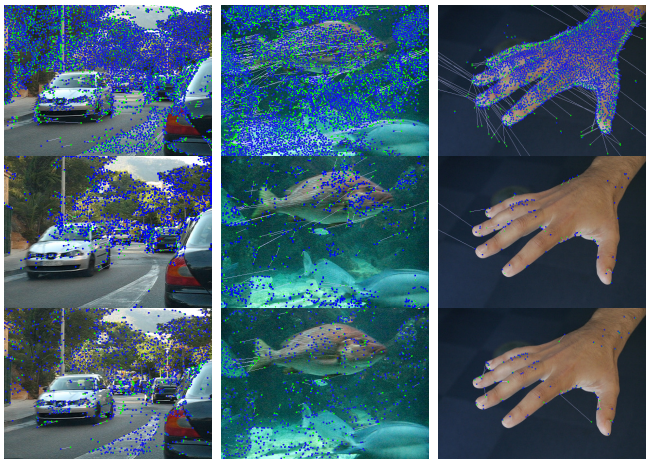

Figure 4. **Top: (a)** KLT tracking results for the MIT sequences. **Middle: (b)** Tracking results by the RLOF**. **Bottom: (c)** Tracking results by MORLOF. Blue rectangles indicate the trajectories endpoint and green circles the corresponding ground truth. Corresponding pairs are connected with light blue lines.

for the sequences are illustrated in Figure 4. The top and middle row shows the behavior of the GFT method, that was parametrised to compute a reduced number of points to track. The trajectory distribution is relocated to highly textured regions. The MORLOF incooperate the texture information too, but by additionally considering the motion clusters information the distribution of the points to track is better balanced. E.g. the road area of sequence camera is sufficient covered by MORLOF trajectories but barely by RLOF** trajectories.

B. Trajectory distribution

To evaluate the proposed motion adaptive feature selection method on a long term basis, we use a 640×480 traffic scenes with about 500 images. These sequences are typical video surveillance sequences with small moving objects. The MORLOF tracker should also be able to select points at significant regions as the moving cars and track them efficiently. To measure the coverage of each object by trajectories, we annotate the contour of each car using the

Figure 5. **Top: (a)** First frame and the annotated motion segmentation of the frame 153 from a static camera. **Bottom: (b)** First frame and the annotated motion segmentation of the frame 153 from a hand-held camera.

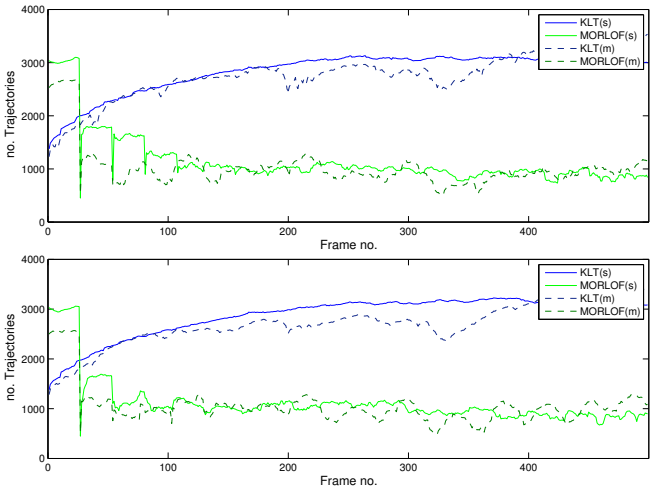

Figure 6. **Top: (a)** Change of the number of trajectories over time for the static sequence (s) and the hand-held captured sequence (m) with CPU implementations. **Bottom: (b)** Change of the number of trajectories over time with the GPU implementations.

segmentation tool provided by Liu *et al.* [19]. Figure 5 shows the first frame of the used sequences and the annotated frame 153. The top sequence was captured using a static camera and the bottom sequence was captured using a moving hand-held camera. This second scene was used in order to proof the robustness of the proposed method with respect to global motion.

The long term evaluation is performed between the MORLOF and the KLT tracker. Table II shows the number of features, i.e. trajectory endpoints, localized at each object . An object id references to the Figures 5(a) and 5(b). Trajectories lying at the background are denoted as BG. The results show that the coverage of the small moving cars are maintained. On average, the number of features per car is decreased slightly with respect to the KLT tracker. The number of background trajectories could be decreased up to 65% in both sequences. Figure 6 shows the change of the number of trajectories over time, which is determined by the number of trajectories lying at the background. The KLT tracker initializes new trajectories until saturation. This is affected by the camera noise that changes the GFT detections. The MORLOF is initialized with a high number of trajectories by the FAST detector. The rejection of background trajectories starts at frame 22 ($\tau_2 = 21$). Followed by reinitializing some of the removed tracks, the system is converging to a stable number of tracks after a few frames.

The execution time of the MORLOF and KLT tracker is shown in Figure 7. In Figure 7(a) the run-time reduction of the CPU implementation is visible, obtaining an overall run-time decrease from 775ms of the KLT tracker to 216ms of the MORLOF for static sequences. Figure 7(b) relativists the direct run-time benefit of saving number of features to track for GPU implementations. Mainly due to the adaptive region strategy of the RLOF, the MORLOF tracker is performing with a lower run-time than the KLT tracker. Beside using a tracker, whose output is a reduced set of trajectories, could be further improve the run-time performance of advanced post-processing techniques, e.g., advanced clustering [2], [1].

V. Conclusions

In this paper we presented an improved tracking method for long term video analysis based on the RLOF method. The proposed tracker takes the motion of the tracked points into account in order to constrain the number of point trajectories needed. Therefore, we use motion cluster oriented feed-back in order to adapt the detector response and remove existing trajectories according to the extracted motion segments. Our experiments with the MIT dataset show that tracking with MORLOF provides an improved accuracy and run-time relation concerning to the KLT and RLOF* tracker. In further experiments we evaluate the proposed method with a traffic sequence about 500 images, where we could show that our method is able reduce the number of trajectories up to 35% with similar trajectory coverage of the foreground objects.

Acknowledgment

The research leading to these results has received funding from the European Communitys FP7 under grant agreement number 261743 (NoE VideoSense).

References

[1] J. S. Katerina Fragkiadaki, "Detection free tracking: Exploiting motion and topology for tracking and segmenting under entanglement," in *Computer Vision and Pattern Recognition (CVPR 11)*, 2011, pp. 2073–2080.

Object	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	BG
MORLOF	0	5	6	4	6	6	9	3	32	9	3	8	11	6	1	31	13	888
KLT	4	9	4	24	14	7	28	10	46	14	7	7	18	13	7	61	17	2544

Object	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	BG
MORLOF	2	3	7	8	6	3	4	15	5	3	15	6	7	0	3	2	3	4	5	4	6	882
KLT	8	5	12	12	12	5	11	13	12	7	17	3	18	1	3	5	11	9	5	8	13	2392

Table II

TOP: (A) EVALUATION OF THE COVERING FOR FRAME 153 OF THE STATIC TRAFFIC SEQUENCE, SEE FIGURE 5(A). **BOTTOM: (B)** EVALUATION OF THE HAND-HELD TRAFFIC SEQUENCE, SEE FIGURE 5(B). THE COVERAGE IS MEASURED BY COUNTING THE NUMBER OF FEATURES LOCALIZED AT EACH OBJECT. BG DENOTES THE NUMBER OF FEATURES LOCALIZED AT THE BACKGROUND.

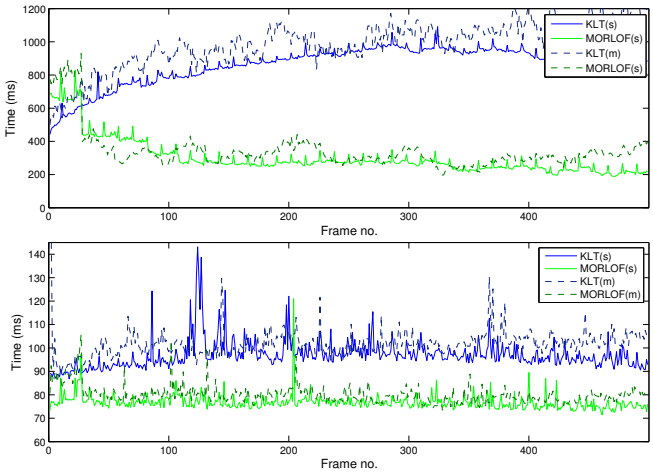

Figure 7. **Top: (a)** Execution time for the static sequence (s) and the hand-held captured sequence (m) with CPU implementations. **Bottom: (b)** Execution time with the GPU implementations.

[2] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *European Conference on Computer Vision (ECCV 10)*, 2010, pp. 282–295.

[3] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," Microsoft Research, Technical Report MSR-TR-2009-179, 2009.

[4] P. Sand and S. Teller, "Particle video: Long-range motion estimation using point trajectories," *Int. J. Comput. Vision*, vol. 80, pp. 72–91, 2008.

[5] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime TV-L^1 optical flow," in *DAGM-Symposium*, 2007, pp. 214–223.

[6] T. Brox, C. Bregler, and J. Malik, "Large displacement optical flow," *Computer Vision and Pattern Recognition (CVPR 09)*, pp. 41–48, 2009.

[7] N. Sundaram, T. Brox, and K. Keutzer, "Dense point trajectories by gpu-accelerated large displacement optical flow," in *European Conference on Computer Vision (ECCV 10)*, 2010, pp. 438–451.

[8] C. Tomasi and T. Kanade, "Detection and tracking of point features," CMU, Technical Report CMU-CS-91-132, 1991.

[9] S. Ali and M. Shah, "A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis," in *Computer Vision and Pattern Recognition (CVPR 07)*, 2007, pp. 1–6.

[10] M. Hu, S. Ali, and M. Shah, "Detecting global motion patterns in complex videos," in *International Conference on Pattern Recognition (ICPR 08)*, 2008, pp. 1–5.

[11] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "Gpu-based video feature tracking and matching," UNC Chapel Hill, Technical Report 06-012, 2006.

[12] T. Senst, V. Eiselein, R. Heras Evangelio, and T. Sikora, "Robust modified L2 local optical flow estimation and feature tracking," in *Workshop on Motion and Video Computing (WMVC 11)*, 2011, pp. 685–690.

[13] C. Zach, D. Gallup, and J. Frahm, "Fast gain-adaptive klt tracking on the gpu," in *Visual Computer Vision on GPUs Workshop*, 2008, pp. 1–7.

[14] T. Senst, B. Unger, I. Keller, and T. Sikora, "Performance evaluation of feature detection for local optical flow tracking," in *International Conference on Pattern Recognition Applications and Methods (ICPRAM 12)*, 2012, pp. 303–309.

[15] L. P. Chew, "Constrained delaunay triangulations," in *Proceedings of the third annual symposium on Computational geometry*, 1987, pp. 215–222.

[16] R. L. Graham and P. Hell, "On the history of the minimum spanning tree problem," *IEEE Annals of the History of Computing*, vol. 7, pp. 43–57, 1985.

[17] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *Transactions on Computers*, vol. 20, pp. 68–86, January 1971.

[18] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision (ECCV 06)*, 2006, pp. 430–443.

[19] C. Liu, W. Freeman, E. Adelson, and Y. Weiss, "Human-assisted motion annotation," in *Computer Vision and Pattern Recognition (CVPR 08)*, 2008, pp. 1–8.