High-Speed Hand Tracking for Studying Human-Computer Interaction

Toni Kuronen[1(✉)], Tuomas Eerola[1], Lasse Lensu[1], Jari Takatalo[2], Jukka Häkkinen[2], and Heikki Kälviäinen[1]

[1] Machine Vision and Pattern Recognition Laboratory (MVPR), School of Engineering Science, Lappeenranta University of Technology (LUT), P.O. Box 20, FI-53851 Lappeenranta, Finland
{toni.kuronen,tuomas.eerola,lasse.lensu,heikki.kalviainen}@lut.fi
http://www2.it.lut.fi/mvpr/

[2] Visual Cognition Research Group, Institute of Behavioural Sciences, University of Helsinki, P.O. Box 9, FI-00014 Helsinki, Finland
{jari.takatalo,jukka.hakkinen}@helsinki.fi
http://www.helsinki.fi/psychology/groups/visualcognition/

Abstract. Understanding how a human behaves while performing human-computer interaction tasks is essential in order to develop better user interfaces. In the case of touch and gesture based interfaces, the main interest is in the characterization of hand movements. The recent developments in imaging technology and computing hardware have made it attractive to exploit high-speed imaging for tracking the hand more accurately both in space and time. However, the tracking algorithm development has been focused on optimizing the robustness and computation speed instead of spatial accuracy, making most of them, as such, insufficient for the accurate measurements of hand movements. In this paper, state-of-the-art tracking algorithms are compared based on their suitability for the finger tracking during human-computer interaction task. Furthermore, various trajectory filtering techniques are evaluated to improve the accuracy and to obtain appropriate hand movement measurements. The experimental results showed that Kernelized Correlation Filters and Spatio-Temporal Context Learning tracking were the best tracking methods obtaining reasonable accuracy and high processing speed while Local Regression filtering and Unscented Kalman Smoother were the most suitable filtering techniques.

Keywords: Hand tracking · High-speed video · Hand trajectories · Filtering · Human-computer interaction

1 Introduction

The motivation for this work comes from the human-computer interaction (HCI) research, and the need to accurately record hand and finger movements of test subjects in various HCI tasks. During the recent years, this has become particularly important due to the rapid development of touch display technology

R.R. Paulsen and K.S. Pedersen (Eds.): SCIA 2015, LNCS 9127, pp. 130–141, 2015.
DOI: 10.1007/978-3-319-19665-7_11

and amount of commercially available touchscreens in smartphones, tablets and other table-top and hand-held devices, as well as, the emergence of different gesture based interfaces. Recording the hand movements can be performed by using hand tracking or general object tracking which has been studied since the 1990s and is an active research area also today [4], [13], [15], [24], [25]. Despite the significant effort, however, the problem of hand tracking cannot be considered solved [9]. From a technical perspective, different robust approaches for hand tracking exist, such as data gloves with electro-mechanical or magnetic sensors that can measure the hand and finger location with high accuracy. However, such devices affect the natural hand motion, are expensive, and hence, cannot be considered a good solution when pursuing natural HCI. As a consequence, there is a need for image-based solutions that provide unobtrusive way to study and track human movement and enable natural interaction between technology.

To accurately record fast phenomena such as reaction times and to robustly track rapid hand movements, high frame rates are needed in imaging. To produce videos with good quality, the high-speed imaging requires more light when compared to imaging with conventional frame rates. Therefore, gray-scale high-speed imaging is in common use making the use of hand tracking methods relying specifically on color information unsuitable. This motivates to apply general object trackers for the problem. In [9], various general object trackers were compared for hand tracking with a primary focus on gray-scale high-speed videos. It was found out that by avoiding the most difficult environments and posture changes, the state-of-the-art trackers are capable of reliable hand and finger tracking.

The main problem in using the existing object tracking methods in accurate measurement of hand and finger movements is that they are developed for applications where high (sub-pixel) accuracy is unnecessary. Instead, the research has focused on developing more computationally efficient and robust methods, i.e., losing the target is considered a much more severe problem than a spatial shift of the tracking window. While these are justified choices in most tracking applications, this is not the case in the hand trajectory measurement in high speed videos where small hand movement between the frames and a controlled environment help to maintain higher robustness, but high accuracy is needed. Even small errors in spatial locations can cause high errors when computing the speed and acceleration. Therefore, the existing tracking algorithms are as such insufficient for the accurate measurements of hand movements and further processing of hand trajectories is required.

In this paper, the work started in [9] is continued by further evaluating an extended set of tracking algorithms to find the best methods for accurate hand movement measurements. Moreover, the earlier work is extended by processing tracked hand trajectories with various filtering techniques. The different methods are evaluated using novel annotated data consisting of high-speed gray-scale videos of a human performing HCI tasks using a touch user interface.

Since the trackers specific for hand tracking rely on color information, the focus of this study is on the state-of-the-art general object trackers. Based on

a literature review and preliminary tracking tests, 12 trackers were selected for further study [14]. These methods are summarized in Table 1.

Table 1. Trackers selected for the experiments

Method	Abbreviation	Implementation
Real-time Compressive Tracking [27]	CT	MATLAB+MEX[1]
Fast Compressive Tracking [28]	FCT	MATLAB+MEX[2]
High-Speed Tracking with Kernelized Correlation Filters [8]	KCF	MATLAB+MEX[3]
Hough-based Tracking of Non-Rigid Objects [5]	HT	C++[4]
Incremental Learning for Robust Visual Tracking [19]	IVT	MATLAB+MEX[5]
Robust Object Tracking with Online Multiple Instance Learning [1]	MIL	MATLAB[6]
Tracking Learning Detection [12]	TLD	MATLAB+MEX[7]
Robust Object Tracking via Sparsity-based Collaborative Model [29]	RSCM	MATLAB+MEX[8]
Fast Tracking via Spatio-Temporal Context Learning [26]	STC	MATLAB[9]
Structured Output Tracking with Kernels [6]	struck	C++[10]
Single and Multiple Object Tracking Using Log-Euclidean Riemannian Subspace and Block-Division Appearance Model [10]	LRS	MATLAB+MEX[11]
Online Object Tracking with Sparse Prototypes [21]	SRPCA	MATLAB[12]

Real-time Compressive Tracking (CT) [27] is a tracking-by-detection method that uses a sparse random matrix to project high-dimensional image features to low-dimensional (compressed) features. The basic idea is to acquire positive samples near the current target location and negative samples far away from the target object at each frame, and use these samples to update the classifier. Then, the location for the next frame is predicted by getting samples from around the last known location and choosing the sample that gets the best classification

[1] http://www4.comp.polyu.edu.hk/~cslzhang/CT/CT.htm
[2] http://www4.comp.polyu.edu.hk/~cslzhang/FCT/FCT.htm
[3] http://www.isr.uc.pt/~henriques/circulant/
[4] http://lrs.icg.tugraz.at/research/houghtrack/
[5] http://www.cs.toronto.edu/~dross/ivt/
[6] http://whluo.net/matlab-code-for-mil-tracker/
[7] http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html
[8] https://github.com/gnebehay/SCM
[9] http://www4.comp.polyu.edu.hk/~cslzhang/STC/STC.htm
[10] http://www.samhare.net/research/struck/code
[11] http://www.iis.ee.ic.ac.uk/~whluo/code.html
[12] http://faculty.ucmerced.edu/mhyang/project/tip13_prototype/TIP12-SP.htm

score. Fast Compressive Tracker (FCT) [28] is an improvement of CT. The speed of the tracker is improved by using a sparse-to-dense search method. First, the object search is done by using a sparse sliding window followed by detection using a dense sliding window for better accuracy.

HoughTrack (HT)[5] is a tracking-by-detection method which is based on the generalized Hough transform. In the method, a Hough-based detector is constantly trained with the current object appearance. Unlike the other selected algorithms, in addition to bounding box tracking, HT outputs also segmented tracking results which is used to limit the amount of background noise supplied to the online learning module.

Incremental learning for robust visual tracking (IVT) [19] learns a low-dimensional subspace representation of the target object and tracks it using a particle filter. Online object tracking with sparse prototypes (SRPCA)[21] is a particle filter based tracking method that utilizes sparse prototypes consisting of PCA basis vectors modeling the object appearance. The main difference to IVT is trivial templates that are applied to handle partial occlusions.

High-Speed Tracking with Kernelized Correlation Filters (KCF) [8] is an improved version of the kernelized correlation filters introduced in [7]. By over-sampling sliding windows, the resulting data matrix can be simplified, the size of the data reduced, and the computation made faster. This can be achieved by taking advantage of Fast Fourier Transform (FFT).

Tracking with online multiple instance learning (MIL) [1] is a tracking-by-detection method that applies the multiple instance learning approach to tracking to account ambiguities in the training data. In the multiple instance learning, positive and negative training examples are presented as sets, and labels are provided for the sets instead of individual instances. By using this approach, updates of the classifier with incorrectly labeled training examples may be avoided and thus, more robust tracking achieved.

Tracking-learning-detection (TLD) [12] is a framework aiming to long-term target tracking by decomposing the task into tracking, learning, and detection sub-tasks. The tracker is tracking the object during the frames whereas the detector localizes all the appearances observed earlier and reinitializes the tracker if required. The final tracker estimate is a combination of the tracker and detector bounding boxes. The third sub-task, learning, tries to estimate the errors of the detector and update it to avoid those in the following frames.

Robust Object Tracking via Sparsity-based Collaborative Model (RSCM) by Zhong et al. [29] contains a sparsity-based discriminative classifier (SDC) and a sparsity-based generative model (SGM). SDC introduces an effective method to compute the confidence value that assigns more weight to the foreground by extracting sparse and determinative features that distinguish the foreground and background better. SGM is a histogram-based method that takes the spatial information of each patch into consideration with an occlusion handling scheme.

Fast Tracking via Spatio-Temporal Context Learning (STC) [26] algorithm works by learning a spatial context model between the target and its surrounding background. The learned model is used to update the spatio-temporal context

model for the following frame. The tracking task is formulated by convolution as a computing task of a confidence map, and the best object location can be estimated by maximizing the confidence map.

The main idea of Structured Output Tracking with Kernels (struck) [6] is to create positive samples from areas containing the object, and negative samples of the background further away from the object. It uses a confidence map and obtains the best location by maximizing a location likelihood function of an object.

Tracker based on Riemannian subspace learning (LRS)[10] is an incrementally learning tracking algorithm that focuses on appearance modeling using a subspace-based approach. The key component in LRS is the log-Euclidean block-division appearance model that aims to adapt to the changes in the objects appearance. In the incremental log-Euclidean Riemannian subspace learning algorithm, covariance matrices of image features are mapped into a vector space with the log-Euclidean Riemannian metric. The log-Euclidean block-division appearance model captures both local and global spatial layout information about the object's appearances. Particle filtering based Bayesian state inference is utilized as the core tracking technique.

2 Trajectory Filtering

In an ideal case, the motion between the frames should be at least one pixel in order to be quantifiable for the trackers. That is not always the case with high-speed videos and can create challenges for the trackers and trajectory analysis. Therefore, filtering of the trajectory data is necessary to obtain accurate velocity and acceleration measurements. Fig. 1 shows an example result of filtering the tracking data.

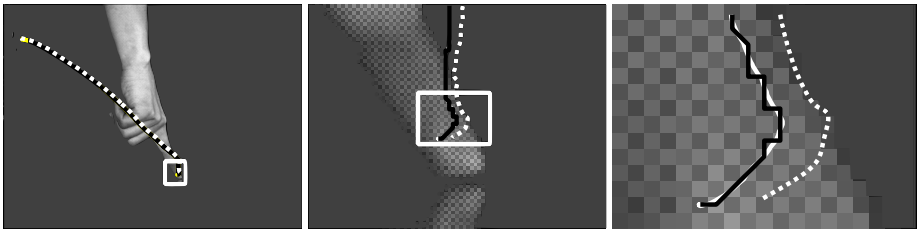


Fig. 1. Raw tracking data (black), the ground truth (dotted white) and filtered tracking data (white)

The following 8 filtering methods were considered in this work: Moving Average (MA) [20], Kalman Filter (KF) [22,23], Extended KF (EKF) [17], Unscented KF (UKF) [11], Local Regression (LOESS) [3], Locally Weighted Scatterplot Smoothing (LOWESS) [3], Savitzky-Golay (S-G) [18], and Total Variation Denoising (TVD) [2] .

MA filter operates by averaging subsets of input data points to produce a sequence of averages. A Kalman filter is an optimal recursive data processing algorithm. EKF is the nonlinear version of the Kalman filter and has been considered as the de-facto standard in nonlinear state estimation. In UKF, unscented transformation is used to calculate the statistics of a random variable which undergoes a nonlinear transformation. It is designed on the principle that it is easier to approximate a probability distribution than an arbitrary nonlinear function. In KF, the predictor predicts parameter values based on the current measurements. The filter estimates parameter values by using the previous and current measurements. The smoothing algorithm estimates the parameter values by using the previous, current, and future measurements: that is, all available data can be used for filtering [23]. Future measurements can be used because the Kalman smoother proceeds backward in time. This also means that the Kalman filter needs to be run before running the smoother.

LOESS and LOWESS were originally developed to enhance visual information on scatterplots by computing and plotting smoothed points by using locally weighted regression. LOESS and LOWESS are methods to estimate the regression surface through a smoothing procedure. S-G is a smoothing filter, also called the polynomial smoothing or least-squares smoothing filter. S-G smoothing reduces noise while maintaining the shape and height of peaks. Total variation (TV) of a signal measures the changes in the signal between signal values. TVD output is obtained by minimizing a TV-based cost function. It was developed to preserve sharp edges in the underlying signal.

3 Experiments

3.1 Data

Data was collected during a HCI experiment where test subjects were advised to perform intentional single finger pointing actions from trigger-box toward a colored target on a touchscreen. The target on the touchscreen was one of 13 objects which formed a circle on the screen, were of different sizes, and lay on different parallaxes. Hand movements were recorded with a Mega Speed MS50K high-speed camera equipped with Nikon Nikkor AF-S 14-24mm F2.8G objective fixed to a 14mm focal length. The camera was positioned on the right side of the test setup, and the distance to the screen was approximately 1.5 meters. The lighting was arranged using an overhead light panel 85 cm above the table surface and 58 cm in depth. The test subject was sitting at the distance of 65 cm from the touch screen and a trigger-box was placed 40 cm away from it.

Dataset contained 11 high-speed videos with 800×600 resolution recorded at 500 fps. Sample frames from the dataset can be seen in Fig. 2. These images illustrate the different end-points of the trajectories. The start-point for all the sequences was the same. The ground truth was annotated manually. Annotations were done for every 5th frame and then interpolated using spline interpolation to get the ground truths for every frame.

Fig. 2. The sample images are from the dataset used in the experiments. Those were all taken from the end point of respective videos. The ground-truth bounding-box can be seen as a white rectangle in the images.

3.2 Results

The tracking experiments were carried out using the original implementations of the authors except in the case of MIL; for that, the implementation by Luo [16] was used. Search area parameters of the trackers were tuned for the video data used, if it was possible with the implementation. For the other parameters, the default values proposed by the original authors were used. The tracking methods were run 10 times for each video and the results were averaged to minimize random factors in tracking. Table 2 shows the results of the trackers for the dataset. The tracking rate of 100% with threshold of 32 pixels center location error was achieved by three of the trackers, KCF being the best one in overall results with the smallest average center location error of 4.65. Also, struck, and STC achieved high accuracy. Length of the videos in total was 10798 frames and individual videos were between 544 and 1407 frames long.

Table 2. Tracking results for Dataset: percentage of correctly tracked frames (TR%) and average center location errors (Err.), and the processing speed (fps). Also, the range of the values from the results are shown. The best results are shown in bold.

Method	TR%	TR% range	Err.	Err. range	fps	fps range
CT	79.43%	0-100%	18.43	3.5-76	99.97	63-121
FCT	17.14%	0-73%	58.74	16-92	118.73	72-150
HT	97.12%	36-100%	15.29	3.1-226	4.65	4-4.9
IVT	74.50%	15-100%	86.75	2.0-448	63.38	51-70
KCF	**100%**	-	**4.65**	1.4-7.4	979.97	728-1236
LRS	20.51%	2-47%	291.32	76-540	8.79	7.8-9.4
MIL	93.82%	24-100%	11.35	2.8-138	0.55	0.4-0.6
RSCM	86.81%	40-100%	18.84	2.2-126	2.50	2.0-2.8
SRPCA	83.64%	24-100%	72.38	1.7-366	10.52	8.3-12.5
STC	**100%**	-	5.13	2.3-6.9	**1291.03**	1156-1330
struck	**100%**	-	4.72	1.6-6.5	118.62	99-153
TLD	68.48%	16-100%	43.55	4.4-139	16.46	8.8-24

When working with high-speed videos, the importance of processing speed is emphasized. The experiments were carried out using a desktop computer with an Intel i5-4570 CPU and 8 GB of memory. The fps measure used in the experiments was calculated without including the image loading times in the calculations to get the raw frame processing speed. The highest fps was measured for STC which showed the best average performance and for KCF which had the peak performance of over 1200 fps. Both achieved processing speeds well over the frame rate of the videos. However, it should be noted that due to the different programming environments (MATLAB, C, etc.) and levels of performance optimization, these results should be considered merely suggestive.

KCF was selected for the further study since it correctly tracked all the frames, had one of the smallest average center location error, was able to process the high-speed videos in real-time. Moreover, earlier tracking experiments [14] have shown that KCF is more robust than STC on diverse video content.

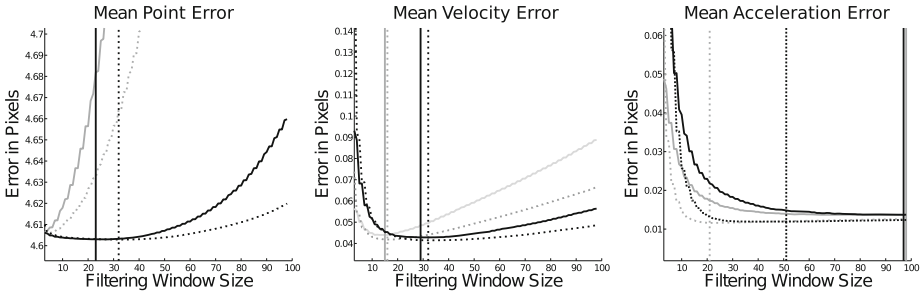
Table 3 summarizes the trajectory filtering results. The results were calculated by averaging the results from all dataset trajectories tracked with KCF tracker. The window size and method parameters were optimized separately for each filtering method. Filtering with Unscented Kalman Smoother (UKS) and TVD are included for comparison. UKS was selected to represent Kalman smoother algorithms since Extended Kalman Smoother and UKS produced similar results. Velocity and acceleration curves for trajectories obtained using Kalman filtering were computed using the Kalman filtering motion model. For the trajectories obtained using other filtering methods, velocity and acceleration curves were computed based on Euclidean distances between trajectory points in consecutive frames.

Table 3. Minimal mean and standard deviations of Position Errors (PE), Velocity Errors (VE), and Acceleration Errors (AE) with different filtering methods. In parentheses is the filtering window size which gave the best result for the filter. The best results are shown in bold.

Error	Moving Average	LOWESS	LOESS	Savitzky-Golay	TVD	UKS	unfiltered
Mean PE	4.6057 (3)	4.6057 (4)	**4.6029** (34)	4.6030 (25)	4.6474	4.6033	4.6099
Mean VE	0.0440 (17)	0.0419 (18)	**0.0415** (34)	0.0428 (31)	0.2052	0.0421	0.2085
Mean AE	0.0137 (83)	**0.0118** (23)	0.0119 (53)	0.0137 (97)	0.3026	0.0125	0.3074
std PE	1.3496 (5)	1.3495 (8)	**1.3459** (38)	1.3461 (29)	1.3860	1.3468	1.3917
std VE	0.0544 (15)	0.0522 (18)	**0.517** (34)	0.0532 (27)	0.2599	0.0526	0.2641
std AE	0.0210 (95)	0.0185 (23)	**0.0185** (39)	0.0206 (85)	0.4599	0.0190	0.4652

From the results shown in Fig. 3, it is obvious that different window sizes were optimal for each derivative of the position. The velocity and acceleration curves needed larger window sizes to get better results than the position. LOESS filtering was the least sensitive to window size with optimal filtering results from the window size range of 34 to 53. The problem with a large window size is that

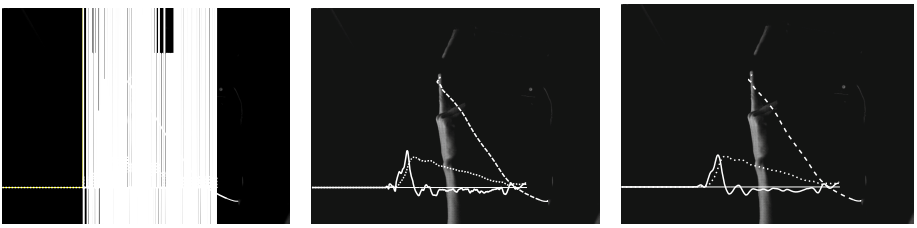
the estimated position starts to drift off from the true position which is very clear in case of moving average and LOWESS filtering.



(a) (b) (c)

Fig. 3. Filtering the effect of the window size on the means of (a) point error; (b) velocity error and (c) acceleration errors. The location of minimum error for each of the methods is indicated with the vertical line. Moving average is shown in grey, LOWESS in dotted grey, LOESS in dotted black, and Savitzky-Golay in black.

An example of how filtering affects the tracking data is shown in Fig. 4. In Fig. 4(a) no filtering is applied to tracking data before calculating velocity and acceleration values. Fig. 4(b) shows the result when position data after tracking is filtered with LOESS filtering with a span of 40 frames, and the velocity and acceleration values are calculated from that filtered data. In Fig. 4(c), also the velocity data is filtered after position data filtering with the same LOESS filtering method. From these results, it is clearly visible that filtering is needed to achieve appropriate velocity and acceleration curves from the tracked hand movement data.



(a) (b) (c)

Fig. 4. Tracking data and velocity and acceleration curves computed from it using: (a) Raw data; (b) Position data filtered with LOESS (span of 40); (c) Position and velocity data filtered with LOESS (span of 40). Trajectory is shown in dashed, velocity in dotted, and acceleration in continuous.

4 Conclusion

In this paper, hand tracking in high-speed videos during HCI tasks, and post-processing of the tracked hand trajectories were studied. The results showed that objects in high-speed video feeds with almost black background can be tracked in real-time with two of the tested trackers. For this research, this meant reaching speeds of over 970 (KCF) and over 1290 (STC) fps on average for the test video sequences which were recorded at 500 frames per second. Thus, the trackers satisfied real-time needs. Even though the performance evaluation for the trackers in this setup did not include the image-loading times, 2.3 milliseconds on average per image with MATLAB, the results are still impressive.

Filtering helps to find smooth acceleration curves to allow us see clearly where the moments of maximum and minimal acceleration are. With appropriate filtering, the velocity and acceleration features of the trajectories got closer to the ground truth. Two filtering methods, LOESS and UKS, produced the most consistent results for all the tests. Selecting one method as the winner raised the question, which one is simpler to use, and that happened to be LOESS. To conclude, with filtering and smoothing the hand-tracking data, it is possible to get to the underlying characteristics of the real movement sequence.

Smoothing the trajectories produced by the trackers gave good results for the derivatives of the position, but sub-pixel accuracy for video sequences which require high precision could be alternative way. By having more accurate positions of the object, one would not need to smooth the trajectories and more accurate results also for the velocities and accelerations of the moving object would be generated. The videos used in this work did not have large scale changes, but adapting to the scale changes on sub-pixel level could help to make the tracking process even more accurate. Also, ground-truth annotation process proved to be a hard undertaking. Clearly visible and accurate marker in test subject's finger would have helped the ground-truth annotation process.

The results provide observations about the suitability of tracking methods for high-speed hand tracking and about how filtering can be applied to produce more appropriate velocity and acceleration curves calculated from the tracking data.

Acknowledgments. The research was carried out in the COPEX project (No. 264429) funded by the Academy of Finland.

References

1. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(8), 1619–1632 (2011)
2. Chambolle, A.: An Algorithm for Total Variation Minimization and Applications. Journal of Mathematical Imaging and Vision **20**(1–2), 89–97 (2004)
3. Cleveland, W.S.: Robust Locally Weighted Regression and Smoothing Scatterplots. Journal of the American Statistical Association **74**(368), 829–836 (1979)

4. Erol, A., Bebis, G., Nicolescu, M., Boyle, R.D., Twombly, X.: Vision-based hand pose estimation: A review. Computer Vision and Image Understanding **108**(1–2), 52–73 (2007). special Issue on Vision for Human-Computer Interaction
5. Godec, M., Roth, P.M., Bischof, H.: Hough-based tracking of non-rigid objects. Computer Vision and Image Understanding **117**(10), 1245–1256 (2012)
6. Hare, S., Saffari, A., Torr, P.H.S.: Struck: structured output tracking with kernels. In: IEEE International Conference on Computer Vision (ICCV), pp. 263–270 (2011)
7. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: Exploiting the circulant structure of tracking-by-detection with kernels. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part IV. LNCS, vol. 7575, pp. 702–715. Springer, Heidelberg (2012)
8. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-Speed Tracking with Kernelized Correlation Filters. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(3), 583–596 (2015)
9. Hiltunen, V., Eerola, T., Lensu, L., Kälviäinen, H.: Comparison of general object trackers for hand tracking in high-speed videos. In: International Conference on Pattern Recognition (ICPR), pp. 2215–2220 (2014)
10. Hu, W., Li, X., Luo, W., Zhang, X., Maybank, S., Zhang, Z.: Single and multiple object tracking using log-Euclidean Riemannian subspace and block-division appearance model. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(12), 2420–2440 (2012)
11. Julier, S.J., Uhlmann, J.K.: A new extension of the kalman filter to nonlinear systems. In: Proceedings of The International Society for Optics and Photonics (SPIE) AeroSense: International Symposium on Aerospace/Defense Sensing, Simulations and Controls (1997)
12. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(7), 1409–1422 (2012)
13. Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., et al.: The visual object tracking VOT2014 challenge results. In: Agapito, L., Bronstein, M.M., Rother, C. (eds.) ECCV 2014 Workshops. LNCS, vol. 8926, pp. 191–217. Springer, Heidelberg (2015)
14. Kuronen, T.: Post-Processing and Analysis of Tracked Hand Trajectories. Master's thesis, Lappeenranta University of Technology (2014)
15. Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A., Hengel, A.V.D.: A Survey of Appearance Models in Visual Object Tracking. ACM Transactions on Intelligent Systems and Technology (TIST) **4**(4), 58:1–58:48 (2013)
16. Luo, W.: Matlab code for Multiple Instance Learning (MIL) Tracker. http://whluo.net/matlab-code-for-mil-tracker/. (accessed: August, 2013)
17. Montemerlo, M., Thrun, S.: Simultaneous localization and mapping with unknown data association using FastSLAM. In: IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 1985–1991 (2003)
18. Orfanidis, S.J.: Introduction to Signal Processing. Prentice Hall international editions, Prentice Hall (1996–2009)
19. Ross, D.A., Lim, J., Lin, R.S., Yang, M.H.: Incremental Learning for Robust Visual Tracking. International Journal of Computer Vision **77**(1–3), 125–141 (2008)
20. Smith, S.W.: The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing (1997)
21. Wang, D., Lu, H., Yang, M.H.: Online Object Tracking With Sparse Prototypes. IEEE Transactions on Image Processing **22**(1), 314–325 (2013)

22. Welch, G., Bishop, G.: An Introduction to the Kalman Filter. Tech. rep. Department of Computer Science, University of North Carolina (1995)
23. Welch, G., Bishop, G.: An Introduction to the kalman filter: SIGGRAPH 2001 course 8. In: Computer Graphics, Annual Conference on Computer Graphics & Interactive Techniques, pp. 12–17 (2001)
24. Wu, Y., Lim, J., Yang, M.H.: Object Tracking Benchmark. IEEE Transactions on Pattern Analysis and Machine Intelligence (2015)
25. Yilmaz, A., Javed, O., Shah, M.: Object Tracking: A Survey. ACM Computing Surveys **38**(4) (2006)
26. Zhang, K., Zhang, L., Liu, Q., Zhang, D., Yang, M.-H.: Fast visual tracking via dense spatio-temporal context learning. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part V. LNCS, vol. 8693, pp. 127–141. Springer, Heidelberg (2014)
27. Zhang, K., Zhang, L., Yang, M.-H.: Real-time compressive tracking. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part III. LNCS, vol. 7574, pp. 864–877. Springer, Heidelberg (2012)
28. Zhang, K., Zhang, L., Yang, M.H.: Fast Compressive Tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **36**(10), 2002–2015 (2014)
29. Zhong, W., Lu, H., Yang, M.H.: Robust Object Tracking via Sparse Collaborative Appearance Model. IEEE Transactions on Image Processing **23**(5), 2356–2368 (2014)