https://doi.org/10.2352/ISSN.2470-1173.2019.13.COIMG-127

Multi-Target Tracking with an Event-Based Vision Sensor and a Partial-Update GMPHD Filter

Benjamin J. Foster*[1,2]*, Dong Hye Ye*[2,3]*, and Charles A. Bouman*[2]*;
[1]***Lockheed Martin Advanced Technology Laboratories; Cherry Hill, NJ, USA;***
[2]***School of Electrical and Computer Engineering, Purdue University; West Lafayette, IN, USA***
[3]***School of Electrical and Computer Engineering, Marquette University; Milwaukee, WI, USA***

Abstract

Tracking a large number of small, similar, high-speed/-agility targets is a challenging problem for current tracking systems that make use of traditional visual sensors. Such targets necessitate very high tracker update rates to keep up with meandering and mutually-occluding target paths. Event-based vision sensors may offer a solution to this problem as they report only "event-based" pixelwise changes in intensity as they occur with a time resolution approaching 1 µs [1], providing data that is much sparser and higher in time-resolution than traditional vision systems. However, this class of sensor presents unique challenges; for example, a single object in the sensor's field of view may produce multiple synchronous or nearly-synchronous events. In addition, performing direct measurement-to-track association for event data on µs to ms timescales introduces problematic computational burdens for scenarios involving large numbers of targets.

The work described in this paper is twofold. We first define and apply an event-clustering procedure to raw events to reduce the amount of data passed into the tracker. This transformation from events to event-clusters provides a) discrimination between event-clusters that correspond to true targets and those that do not and b) reduction in tracking computation time. Second, we define and apply a partial-update Gaussian mixture probability hypothesis density (GMPHD) filter [2] for tracking using event-cluster data. We demonstrate increased computational performance over the standard GMPHD filter while achieving comparable tracking performance per the optimal sub-pattern assignment (OSPA) metric [3].

Introduction

Visual tracking of individual members of large groups has been studied across a number of application areas ranging from analysis of human crowds [4] to bat population surveys [5]. Kinematic information extracted from multitarget tracking (MTT) algorithms also provides immense value for commercial and defense applications. For example, consider the case of tracking members of an airborne swarm: kinematic information about the swarm's members can enable defensive action against swarms as well as inference about swarm structure and classification of swarm behaviors.

One of the primary challenges of MTT problems is the uncertainty of correspondence between measurement and target due to missed detections, extraneous measurements, and variation in the number of targets in the scene. Solving this problem increases in difficulty when the targets are similar in appearance and are high-speed/-agility. State estimation algorithms (e.g. the Kalman filter and its variants) degrade in performance when the associated motion model is inconsistent with true target dynamics; if the tracker update rate is not sufficient for keeping up with the dynamics of the target(s), tracking performance suffers substantially. In addition, processing full-frame data at high framerate is computationally intensive and may push computation time or system size, weight, and power (SWaP) to intolerably high levels.

The use of event-based vision sensors in place of traditional "frame-based" sensors offers a favorable combination of increased update rate and decreased data rate, potentially suiting them for low-SWaP MTT systems. Event-based cameras feature independent pixels which only send information in response to pixel-wise brightness changes in the scene as they occur. As such, the output is a stream of asynchronous events which consist of its spatial coordinates (x, y), a timestamp t, and a binary polarity p. A positive polarity indicates that the event was generated by an increase in intensity at the pixel, while a negative polarity indicates that the event was generated by a decrease in intensity at the pixel.

There exist multiple lines of research involving tracking with event-based vision sensors [6] [7], including those that examine the problem of tracking a large number of similarly-shaped objects [8]. However, these studies employ tracking methods which are dependent on explicit or partial measurement-to-track association which are computationally burdensome. Random finite set (RFS) methods such as the probability hypothesis density (PHD) filter and its variants avoid explicit measurement-to-track association altogether, providing an alternative that can be more easily worked into low-SWaP, real-time tracking applications. In addition, RFS methods propagate a set-valued state vector and estimate the cardinality of the target set an the multitarget state. This is useful for swarm tracking applications as the number of individuals in the swarm may be dynamic. In this paper, we apply a modified GMPHD filter for tracking of clusters of events.

The GMPHD filter propagates an intensity map of the entire space at every timestep. However, areas of the map that are not measurement-rich may not change significantly at every timestep, so updating these areas at every timestep is extraneous. We introduce a heuristic for updating only measurement-rich areas of the GMPHD intensity map in order to reduce computation time; we show that the performance of this partial-update GMPHD filter is comparable to that of the full update filter.

Event-Based Data Generation

Though our group possesses an event-based camera, the assembly and ground-truthing of an actual multi-agent swarm proved to be outside of the scope of this effort. Instead, we used the event camera simulator Blender plugin described in [9] to generate our dataset. The dataset consists of two groups of 10 UAVs moving from right to left across the simulated sensor's field of view. Tree objects that periodically sway slightly right and left are also present in the foreground of the dataset in order to induce clutter event measurements. In addition to generating events with μs time resolution, we also extract groundtruth locations of targets with 1 ms time resolution for performance assessment purposes. A screen capture of our dataset is displayed in Fig. 1.

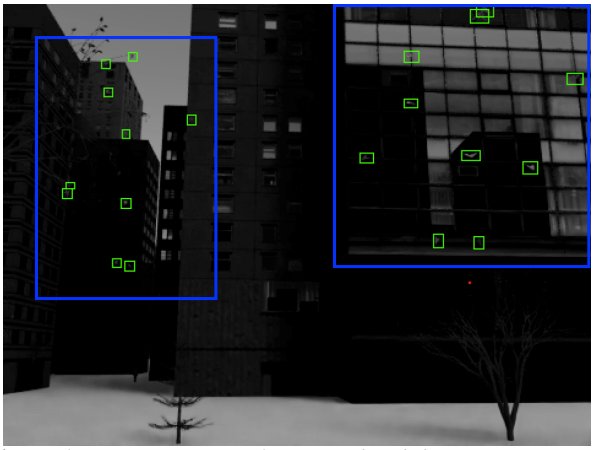


Figure 1: *Screen capture of our simulated dataset. Note simulated tree objects in the foreground; movement of leaves generates clutter measurements in our scenario. Green boxes denote groundtruth position of simulated UAV targets. The above image is full-frame for visualization purposes but we use corresponding events generated by the event camera simulator Blender plugin [9] in our analysis.*

Event Clustering

Raw event-based vision sensor data is a poor choice as a direct input into our tracking system because a) passing every event directly into the tracker quickly becomes computationally burdensome and b) a single object in the sensor's field of view may produce multiple synchronous or nearly-synchronous events. Our approach to addressing this problem involves applying a prefiltering step that clusters multiple events into a single "event-cluster" object. We then prune extraneous event-clusters using the variance of spatial location of clustered events as a feature to discriminate true detections from false alarms. This section details the operation of our event-clustering and pruning algorithm.

Let e denote a single event and c represent an event-cluster, respectively. We also denote pruned event-clusters s, which are ultimately fed into the GMPHD filter as input. Each of these variables can be parameterized as follows:

$$e \leftarrow (t,x,y,p) \tag{1}$$

$$c \leftarrow (t,x,y,N) \tag{2}$$

$$s \leftarrow (t,x,y) \tag{3}$$

In (1), t, x, y, and p represent the time stamp, horizontal and vertical coordinates, and the polarity of the event, respectively. In (2), t, x, y, and N represent the time stamp, the event-cluster horizontal and vertical coordinates, and the number of events assigned to the cluster, respectively. In (3), t represents the timestep when the cluster is reported, while x and y represent the corresponding spatial location of the reported cluster. Let C denotes the collection of event-clusters c.

Algorithm 1

procedure CLUSTER&PRUNE
  Input: $e, C=\{c_1,...,c_i,...,c_M\}$; *Output* : s, C
  if new event occurs **then**
    $i^*=argmin_i dist(c_i,e)$
    if $dist(c_{i^*},e) < T_d$ **then**
      $c_{i^*} \leftarrow UpdateCluster(c_{i^*},e)$
      if $c_{i^*}.N \geq T_N$ **then**
        $C \leftarrow DeleteCluster(C,c_i^*)$
        if $TestCluster(c_{i^*}) = 1$ **then**
          $s \leftarrow ReportCluster(c_{i^*})$
          $C \leftarrow DeleteOldCluster(C,e)$
    else
      $C \leftarrow AddCluster(C,e)$

Algorithm 1 specifies the procedure of clustering and pruning on event-based data. For each raw received event e, we find its nearest neighboring event-cluster from the collection of event-clusters C. If the nearest neighbor event-cluster is within a threshold distance T_d, we update the nearest neighbor event-cluster with location of the input event. Specifically, we update the spatial location of event-cluster with the weighted summation between event location and event-cluster location. We also update the variance of spatial distribution of events assigned to the event-cluster according to:

$$c.V_x^+ \leftarrow \frac{c.N}{c.N+1}\{c.V_x^+ + \frac{(c.x-e.x)^2}{c.N+1}\} \tag{4}$$

$$c.V_y^+ \leftarrow \frac{c.N}{c.N+1}\{c.V_y^+ + \frac{(c.y-e.y)^2}{c.N+1}\} \tag{5}$$

$$c.V_x^- \leftarrow \frac{c.N}{c.N+1}\{c.V_x^- + \frac{(c.x-e.x)^2}{c.N+1}\} \tag{6}$$

$$c.V_y^- \leftarrow \frac{c.N}{c.N+1}\{c.V_y^- + \frac{(c.y-e.y)^2}{c.N+1}\} \tag{7}$$

where V_x^+, V_y^+, V_x^- and V_y^- represent the variance of spatial distribution of positive and negative events associated with event-cluster c, respectively. We use an online algorithm to compute the variance without keeping all events in the event-cluster for computational and memory efficiency.

Next, if the number of events associated with c increases above a threshold number T_N, we remove c from the intermediate cluster set C and test the event-cluster based on the variance of spatial distribution of clustered events. If the variance of this event-cluster is below a threshold T_v, we report the event-cluster to the tracker. In the *DeleteOldCluster* subroutine, we

delete the event-clusters which are not updated for a period exceeding a threshold T_t. If the nearest neighbor event-cluster is too far away, we create a new event-cluster and add it into the existing event-cluster set C. Though this procedure is similar to previous approaches [6], testing the spatial variance of the cluster before reporting provides additional robustness to noise generated at the sensor.

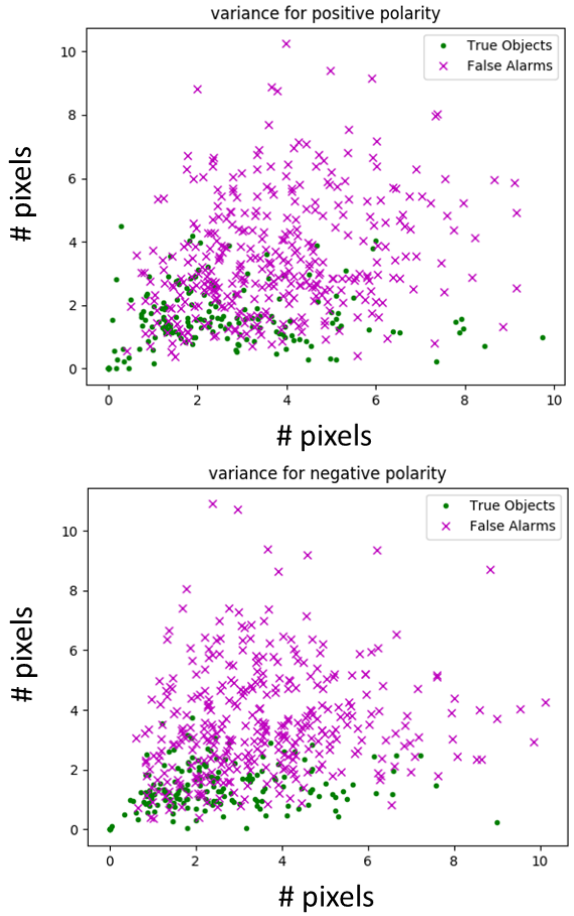


Figure 2: *Two figures show the spatial distribution of positive (top) and negative (bottom) events for each event-cluster in variance feature space. Most true objects have small variance and accumulate in the bottom left side. Based on these two plots we can set thresholds to prune out many false alarms. Units of all axes are "number of pixels"*

Fig. 2 displays the variance of spatial distribution of positive and negative events for each cluster. The green dots and purple crosses represent true objects and false alarms, respectively. We observe that green dots (which correspond to true objects) are generally located nearer to the (0,0) corner of the plot. This corresponds to our hypothesis that events for true objects occur near object centers, which will in turn result in small variances in spatial distribution (we note that this assumption is motivated by the fact that objects are small relative to pixel size). In addition, we separate the events based on the polarity into positive and negative groups. This increases the dimension of our feature space, improving discrimination power.

Table 1: *Event statistics of i) raw events, ii) clustered events, and iii) clustered and pruned events for 10 second simulated data scenario. Note that the total number of reports is greatly reduced by employing the clustering procedure and the number false alarms is further reduced by the pruning procedure.*

	Raw Data	Clustering Only	Clustering & Pruning
# of input events to tracker	196311	3847	2152
# of input events from true targets	82829	1726	1203
# of input events from false targets	113482	2119	949

In Table 1, we report the number of input events to the GM-PHD filter when operating on (i) raw data, (ii) event-clusters without pruning, and (iii) event-cluster with pruning when analyzing the 10 second simulated dataset. After forming event-clusters, the number of input events to GMPHD is significantly reduced from 196311 events to 3847 event-clusters, a 50-fold reduction. When we also apply our pruning step to the list of event-clusters, we reject about 55.2% of false alarms while preserving about 69.7% of clusters corresponding true moving objects. Note that before pruning the number of false alarms significantly exceeds the number of "true" clusters, while after pruning the number of "true clusters significantly exceeds the number of false alarms. This balance can be tuned by adjusting event-clustering and pruning parameters as desired.

The output of our Algorithm 1 is stored as a list. In order to visualize the results of our preprocessing method, we integrate part of the data into one frame (between 1100ms and 1400ms) and overlay it on a frame capture by a standard camera (also simulated in Blender) as illustrated in Fig. 3.

The GMPHD Filter

The PHD filter described in [10] propagates a joint estimate of the multitarget state and cardinality via the intensity $v_k(x)$. This is the first order moment of the multi-target posterior $p_k(X_k|Z_{1:k})$, which is analogous to the single-target posterior propagated in the canonical Kalman filter. Computing the exact PHD recursion is generally computationally intractable, so approximations such as Vo and Ma's GMPHD filter [2] are often applied in practice. We briefly outline the operation of the GMPHD filter as described in [2].

In this formulation, the first-order moment of the PHD, or "intensity", for newly-birthed targets is specified as a Gaussian mixture given by:

$$\gamma_k(x) = \sum_{i=1}^{J_{\gamma,k}} w_{\gamma,k}^{(i)} \mathcal{N}\left(x; m_{\gamma,k}^{(i)}, P_{\gamma,k}^{(i)}\right) \tag{8}$$

where $\mathcal{N}(\cdot; m, P)$ is a Gaussian density parameterized by mean m and covariance P, $w_{\gamma,k}^{(i)}$ is the weight of the ith Gaussian component of the birth intensity, and $J_{\gamma,k}$ is the number of Gaussian components that comprise the birth intensity.

The posterior intensity is comprised of the predicted intensity from the previous step plus the birth intensity (other formulations provide for "spawning" of targets from existing targets, but

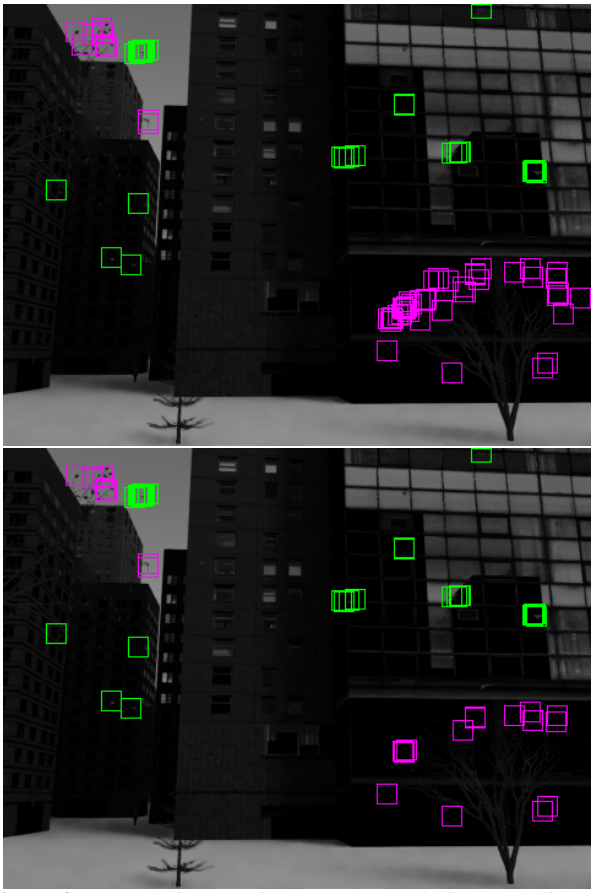


Figure 3: *Reported event-clusters across a 300ms period overlaid on a full-frame capture of our dataset. Green and magenta boxes represent reported event-clusters corresponding to true objects and false alarms, respectively. Top: reported event-clusters without spatial-variance-based pruning. Bottom: reported event-clusters after spatial-variance-based pruning. Note that false alarm event-clusters are significantly reduced by pruning.*

that is not examined in this work).

$$v_{k|k-1}(x) = v_{S,k|k-1}(x) + \gamma_k(x) \tag{9}$$

Note that the predicted intensity is also specified as a Gaussian mixture:

$$v_{S,k|k-1}(x) = p_S \sum_{i=1}^{J_k} w_k^{(i)} \mathcal{N}\left(x; m_{S,k|k-1}^{(j)}, P_{S,k|k-1}^{(j)}\right) \tag{10}$$

where probability of target survival is given by p_S, and is assumed to be constant. The posterior intensity is then comprised of the predicted intensity and $v_{D,k}(x,z)$, where:

$$v_k(x) = (1-p_D)v_{k|k-1}(x) + \sum_{z\in Z_k} v_{D,k}(x;z) \tag{11}$$

and:

$$v_{D,k}(x;z) = \sum_{j-1}^{J_{k|k-1}} w_k^{(j)}(z)\mathcal{N}\left(x; m_{k|k}^{(j)}(z), P_{k|k}^{(j)}\right) \tag{12}$$

$$w_k^{(j)}(z) = \frac{p_D w_{k|k-1}^{(j)} q_k^{(j)}(z)}{\kappa(z) + p_D \sum_{l=1}^{J_{k|k-1}} w_{k|k-1}^{(l)} q_k^{(l)} z} \tag{13}$$

$$q_k^{(j)} = \mathcal{N}\left(z; H_k m_{k|k-1}^{(j)}, R_k + H_k P_{k|k-1}^{(j)} H_k^T\right) \tag{14}$$

$$P_{k|k}^{(j)} = \left[I - K_k^{(j)} H_k\right] P_{k|k-1}^{(j)} \tag{15}$$

$$K_k^{(j)} = P_{k|k-1}^{(j)} H_k^T \left(H_k P_{k|k-1}^{(j)} H_k^T + R_k\right)^{-1} \tag{16}$$

It is instructive to note that Eqs. 15 and 16 are the Kalman update equations. In addition, we estimate the number of targets in the scene by summing the weights of the Gaussian components that comprise the posterior intensity. The number of these Gaussian components can grow very large as the filter iterates across many timesteps; as such, heuristic pruning procedures are employed to mitigate the computational burden associated with maintaining this large list of components [2].

Applying a Partial-Update GMPHD Filter to Event-Based Data

In order to mitigate computational burden associated with updating the entire GMPHD intensity with 1 ms time resolution we adopt the heuristic that, in general, the GMPHD update is only computed in regions in the measurement-space neighborhood of the reported events at the current timestep. The entire intensity function is updated periodically, but this full-update period p is chosen to be at least an order of magnitude larger than the time resolution of event updates (i.e. $p > 10$). Values of p that are too small provide little computational benefit, while values of p that are too large do not allow the filter to maintain extant Gaussian components of the intensity in regions of the measurement space where measurements are sparse.

In order to implement the partial-update GMPHD filter, we choose a sector size s, then divide the measurement space $\mathscr{M}$ into n_s sectors of size s, generating a list of n sectors $\mathscr{S}_k$. We then match the list of reported event-clusters C_k against $\mathscr{S}_k$. If an event-cluster is reported in sector $\mathscr{S}_k$, then Gaussian components of the intensity whose means $m_k^{(i)}$ fall within $\mathscr{S}_{v,k}$ are updated normally as in [2]; otherwise if no event-clusters are reported in $\mathscr{S}_k$ it is not updated at the current timestep k.

The choice of sector size greatly influences the performance of the partial-update GMPHD filter. If the sector size is chosen to be very small, then the procedure of associating sectors with event-clusters becomes computationally burdensome and can even eclipse the computational burden of direct measurement-to-target association; this defeats much of the utility of RFS formulation of the problem. However, a sector size that is too large may approach the full size of the measurement space. In the case where the sector size is equivalent to the size of the measurement space, the partial-update GMPHD filter must update the entire intensity for every measurement - this is just the standard GMPHD filter.

The entirety of the partial-update procedure is outlined in Algorithm 2.

Algorithm 2

procedure PARTIALUPDATE_GMPHD_FILTER

Inputs: GM component parameters $\{w_{k-1}^{(i)}, m_{k-1}^{(i)}, P_{k-1}^{(i)}\}$
Reported Event Clusters $C_k : \{c_1, c_2, ..., c_N\}$
Active Sector List $\mathscr{S}_k$
Update Period p

for each timestep k **do**
  GMPHD_PREDICTIONFORBIRTHTARGETS[2]
  for $i \in J_{k-1}$ **do**
    GMPHD_PREDICTIONFOREXISTINGTARGET[2]
  if mod $(k, p) == 0$ **then**
    for $i \in J_{k-1}$ **do**
      GMPHD_UPDATE[2]
  else
    for $i \in J_{k-1}$ **do**
      if GETSECTOR$(m_{k-1}^{(i)}) \in \mathscr{S}_k$ **then**
        GMPHD_UPDATE[2]

  GMPHD_PRUNE[2]

Results

We investigate the performance of the GMPHD filter and the partial-update GMPHD filter on the output of the event-clustering procedures described earlier in this work. We use optimal subpattern assignment (OSPA) [3] to measure performance of the GMPHD and partial-update GMPHD filters. OSPA captures the ability of a multitarget filter to correctly estimate the number of targets in the scene as well as the states of these extant targets; a high OSPA value is indicative of poor performance, while a low OSPA value is indicative of good performance (the minimum possible OSPA value is 0).

In order to demonstrate the utility of both the event-cluster pruning procedure and the partial-update GMPHD filter, we perform four experiments: we first input our raw event data into i) the event-clustering procedure without pruning and ii) the event-clustering procedure with pruning. We then feed both of these event-cluster output into the full-update GMPHD filter and the partial-update GMPHD filter (we choose the partial-update GMPHD filter's full-update period to be p=20 and the sector size to be s = 60x60 = 3600 pixels). As such, we generate four multi-target filtering outputs: unpruned event-clusters filtered by the full-update GMPHD filter, unpruned event-clusters filtered by the partial-update GMPHD filter, unpruned event-clusters filtered by the partial-update GMPHD filter, and pruned event-clusters filtered by the partial-update GMPHD filter.

Runtimes of the four experiments are displayed in Table 2, while OSPA values over time corresponding to full- and partial-update GMPHD filter experiments on pruned event-cluster data are pictured in Figure 4. We observe in Table 2 that computational performance is improved significantly when we employ the event-cluster pruning procedure and improves slightly but measurably when we employ the partial-update GMPHD filter. In addition,

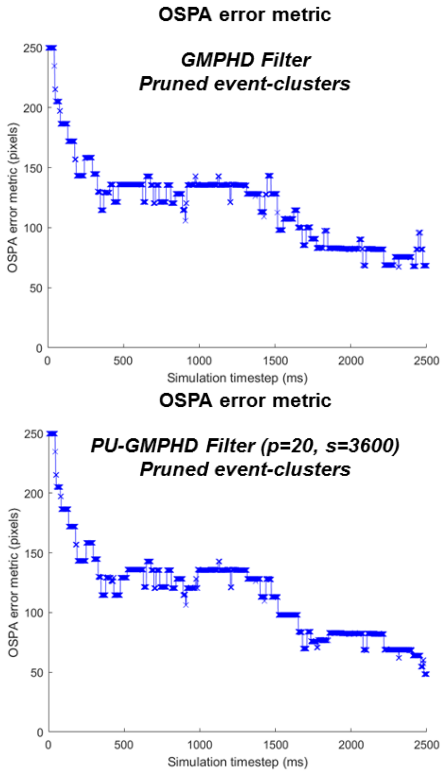


Figure 4: *Top: OSPA performance of full-update GMPHD filter on pruned event-cluster data. Bottom: OSPA performance of partial-update GMPHD filter on pruned event-cluster data. Note that performance is comparable between the two methodologies despite the partial-update GMPHD filter being substantially less computationally burdensome.*

we observe in Figure 4 that multi-target filtering performance of the partial-update GMPHD filter mirrors that of the full-update GMPHD filter despite the reduction in computation time depicted in Table 2.

Table 2: *Runtimes for: Full-update GMPHD filter + pruned event-clusters (top left); Full-update GMPHD filter + unpruned event-clusters (bottom left); Partial-update GMPHD filter + pruned event-clusters (top right); and Partial-update GMPHD filter + unpruned event-clusters (bottom right). Parameters used with the partial-update GMPHD filter were p = 20, s = 60x60 = 3600 pixels.*

	Full-Update GMPHD Filter	Partial-Update GMPHD Filter
Runtime w/ prune	17.14s	16.48s
Runtime w/o prune	22.96s	21.10s

Conclusion and Future Work

In order to filter the significant noise associated with event-based image sensor measurements, we employ an event-clustering procedure that prunes event-clusters with either high spatial vari-

ance or a period of inactivity that exceeds a threshold T_t. Once we have performed the event-clustering step, we pass event-clusters into the GMPHD filter and perform tracking. Additionally, we define a partial-update GMPHD filter that only partially updates the intensity according to the locations of reported clusters to mitigate computational burden. It is demonstrated that performance of the partial-update GMPHD filter is comparable to that of the full-update filter while reducing computation requirements.

Future work will include testing against high-clutter scenarios and building upon the heuristic partial update rule to include more sophisticated update rules developed in the sensor selection literature [11]. In addition, a more detailed analysis of the computational burden associated with the method is necessary for assessing feasibility for low-SWaP systems.

References

[1] R. Berner, C. Brandli, M. Yang, S.C. Liu, and T. Delbruck, A 240x180 120dB 10mW 12us-latency sparse output vision sensor for mobile applications, Proceedings of the International Image Sensors Workshop, pp. 41-44, 2013.

[2] B.N. Vo and W.K. Ma, The Gaussian mixture probability hypothesis density filter, IEEE Trans. on Signal Processing, 54, 11, pp. 4091-4104, 2006.

[3] D. Schuhmacher, B.T. Vo, and B.N. Vo, A consistent metric for performance evaluation of multi-object filters, IEEE Trans. on Signal Processing, 56, 8, pp. 3447-3457, 2008.

[4] M. Rodriguez, S. Ali, and T. Kanade, Tracking in unstructured crowded scenes, International Conference on Computer Vision, pp. 1389-1396, 2009.

[5] M. Betke, D. E. Hirsh, A. Bagchi, N.I. Hristov, N. C. Makris, and T.H. Kunz, Tracking large variable numbers of objects in clutter, IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-8, 2007.

[6] M. Litzenberger, C. Posch, D. Bauer, A.N. Belbachir, P. Schon, B. Kohn, and H. Garn, Embedded vision system for real-time object tracking using an asynchronous transient vision sensor, Digital Signal Processing Workshop, 12th-Signal Processing Educational Workshop, pp. 173-178, 2006.

[7] D.R. Valeiras, X. Lagorce, X. Clady, C. Bartolozzi, S.H. Ieng, and R. Benosman, An asynchronous neuromorphic event-driven visual part-based shape tracking, IEEE Trans. on Neural Networks and Learning Systems, 26, 12, pp. 3045-3059, 2015.

[8] Z. Ni, A. Bolopion, J. Agnus, and R. Benosman, Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics, IEEE Trans. on Robotics, 28, 5, pp. 1081-1089, 2012.

[9] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM, The International Journal of Robotics Research, 36, 2, pp. 142-149, 2017.

[10] R. P. S. Mahler, Multitarget Bayes filtering via first-order multitarget moments, IEEE Trans. on Aerospace and Electronic Systems, 39, 4, pp. 1152-1178, 2003.

[11] A. O. Hero, and D. Cochran, Sensor management: Past, present, and future, IEEE Sensors Journal, 11, 12, pp. 3064-3075, 2011.

Author Biography

Benjamin Foster received his BA in physics from Wabash College (2012) and his MSECE from Purdue University (2014). He is a Senior Member of the Engineering Staff at Lockheed Martin Advanced Technology Laboratories in Cherry Hill, NJ and is also a PhD student at Purdue University. His research interests include inverse problems, machine learning, image processing, and acoustic signal processing.

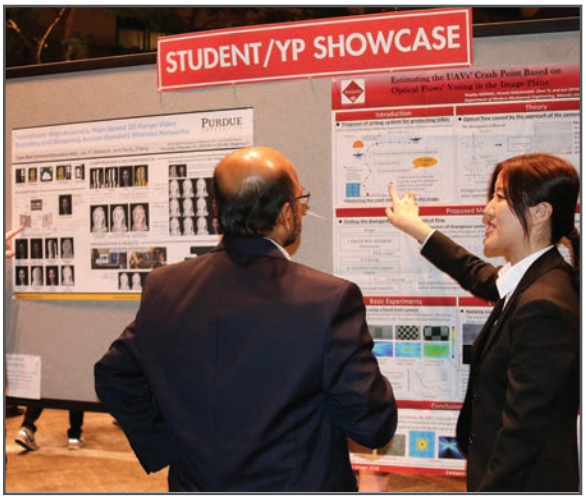
JOIN US AT THE NEXT EI!

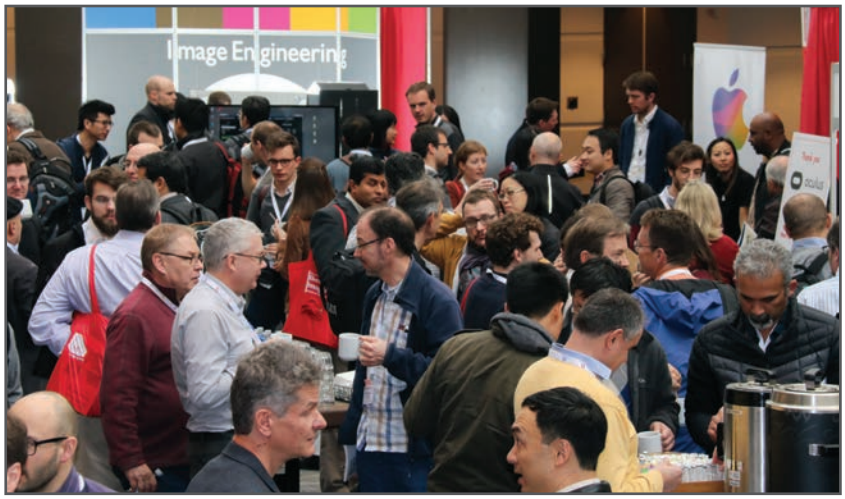
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org