2011 15th International Conference on Information Visualisation

Visualization of Automated and Manual Trajectories in Wide-Area Motion Imagery

Anoop Haridas, Rengarajan Pelapur, Joshua Fraser, Filiz Bunyak, Kannappan Palaniappan
Department of Computer Science, University of Missouri
Columbia, MO 65201 USA
{ahkrc,rvpnc4}@mail.mizzou.edu, jbfraser@gmail.com, {bunyak,palaniappank}@missouri.edu

Abstract

The task of automated object tracking and performance assessment in low frame rate, persistent, wide spatial coverage motion imagery is an emerging research domain. The collection of hundreds to tens of thousands of dense trajectories produced by such automatic algorithms along with the subset of manually verified tracks across several coordinate systems require new tools for effective human computer interfaces and exploratory trajectory visualization. We describe an interactive visualization system that supports very large gigapixel per frame video; facilitates rapid, intuitive monitoring and analysis of tracking algorithm execution; provides visual methods for the intercomparison of very long manual tracks with multi-segmented automatic tracker outputs; and a flexible KOLAM TrackingSimulator (KOLAM-TS) middleware that generates visualization data by automating the object tracker performance testing and benchmarking process.

I. Introduction

Tracking in wide-area motion imagery is a challenging research domain that is receiving a lot of current interest. The visualization of hundreds to thousands of tracks resulting from automated and manual tracking of objects offers new challenges in visualization and visual analytics. Even with standard video sequences, also known as full motion video (FMV), meaningful comparisons of tracking algorithm behavior with quantitative performance metrics were difficult to perform due to the paucity of standard video datasets with associated manual labeled ground truth. However, recent work has led to the creation of extensive, open repositories of *non-wide area* video datasets, tools and ground truth. Notable examples include open source tools such as the Video Performance Evaluation Resource (ViPER-GT) of scripts and Java programs [4], which allows for metadata viewing and editing, ground truth generation and annotation of video including a frame accurate MPEG-1 decoder; VIPER-PE a scriptable command-line based performance evaluation tool; VirtualDub for frame accurate capturing, playing back and filtering of video in AVI format; the Video Surveillance Online Repository (ViSOR) project [24], which comprises a web-based dynamic, shareable, open repository of surveillance video sequences and annotations using an event ontology; and the Scoring, Truthing And Registration Toolkit (START), for semi-automated ground truth generation using a keyframe approach [20]. Several workshops, such as the PETS series and the VSSN series, and national-level projects, such as i-LIDS [7] and ETISEO [11], utilize the ViPER-XML annotation format in their video databases. Another important example is the ground truth motion database developed along with the layer segmentation and motion annotation tools at MIT CSAIL [9]. Since manual ground truth creation from real video is time consuming and error-prone an alternative approach using computer graphics tools to automatically create precise ground truth from realistic/synthetic video of virtual worlds using simulated cameras has also been studied [12], [18], [19]. The NGA coordinated Motion Imagery Standards Board (MISB) has been developing a metadata architecture and standards-based software interfaces for Video Moving Target Indicator (VMTI) systems to analyze and share activity-based GeoINT and tracking results for characterizing actions and interactions in a wide range of motion imagery including FMV and Large Volume Streaming Data (LVSD) [21].

The term LVSD is a NATO designation for a class of imagery also referred to as wide-area persistent surveillance (WAPS), wide-area aerial surveillance (WAAS), wide-area large format (WALF) and WAMI, which is an emerging area of interest, due to its large scale continuous coverage of geospatial regions for a variety of applications [16], [2], [17]. Given the established importance of FMV annotation ground truth databases and performance metrics, increasing attention is being focused on developing equivalent capability for WAMI. Working with WAMI data presents a unique set of challenges, distinct from standard video surveillance data, including interactive visualization of very large time-sequence imagery (100 Mpixels to 10 Gpixels per frame at rates of one Hz or faster using images from an array of smaller cameras with VIS, IR, MSI or HSI sensors), efficient algorithms for mosaicing, georegistration, stabilization, multi-target tracking, event and activity analysis [16] [17]. The difficulties of tracking vehicles in low frame rate aerial wide-area motion imagery are many, including large object displacements, parallax and occlusions from tall structures, low contrast, moving seams across camera boundaries, significant object appearance changes with viewing direction, shadows and are described in recent publications including [16], [13], [3], [17], [22], [8]. Unlike regular video surveillance databases, there are currently only a few WAMI datasets that are available in the public domain. One example is the PSS imagery described in this paper; another is the Columbus Large Image Format (CLIF) dataset, collected in a flyover of the Ohio

1550-6037/11 $26.00 © 2011 IEEE
DOI 10.1109/IV.2011.67

State University Campus in October 2007 [1]. Visualizing the statistical distribution of a dense set of automatically estimated tracks in a limited geographical region by overlaying a large number of trajectories is described in [8], [22].

The WAMI database utilized here for visualization, tracking and ground truth generation purposes, is from the Persistent Surveillance Systems (PSS) event management, law enforcement and emergency response collection. WAMI datasets were collected using an eight camera array on an airborne platform producing 256 megapixel mosaiced georegistered images. We used sample WAMI consisting of several thousand frames collected over Philadelphia (March 13, 2008) for this paper. Some of the features of the Philadelphia WAMI dataset are listed in Table I.

Frame Rate:	1 frame per second (fps)
Altitude:	3,500 – 12,500 ft.
Coverage:	4 square miles, 80°x 60°fov
GSD:	25 – 50 cm
Pixel Type:	Grayscale
Bandwidth:	1 TB/hr; 16K x 16K pixels / frame
File Format:	Tiled JPEG pyramids

TABLE I
WAMI NORTH PHILADELPHIA, PENNSYLVANIA DATASET CHARACTERISTICS (DATASETS COURTESY OF PSS).

We have summarized some of the challenges in interactive visualization of WAMI datasets. The rest of the paper is organized as follows. First, we briefly describe our visualization platform KOLAM, followed by details about KOLAM-TS which automates tracker execution and performance evaluation. Then the different parser modules developed to handle the various ground truth formats are explained and the need for a standard format for ground truth described followed by conclusions.

II. KOLAM INTERACTIVE VISUALIZATION TOOL

The KOLAM (K-tiles for Optimized muLtiresolution Access with coMpression) application was developed using C++ and the Qt 4.x SDK to provide cross-platform (Windows, Mac OS X, Linux) interactive visualization of extremely large, time-varying image datasets, on the order of hundreds of gigabytes to terabytes in size. KOLAM achieves smooth interactive display, navigation and analysis of massive datasets through a combination of pyramidal out-of-core data structures, efficient application-level paging, and concurrent I/O and processing [14], [15]. An early motivation for developing KOLAM was to extend linked multiple window visualization tools [5], [6], [23] to support extremely large datasets. Other tools for visualization of WAMI datasets include Persistent Surveillance Systems (PSS) iView and MIT Lincoln Laboratory Advanced Persistent Image eXploitation (APIX) Viewer and processing system.

The pyramidal data structure used for organizing the image into spatially coherent multiresolution tiles using a hierarchical scale–and–tile architecture is shown in Figure 1. Each successive image layer is scaled by one-fourth of the prior

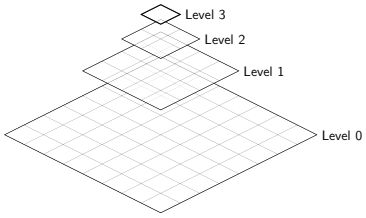

Fig. 1. Multiresolution tiled image pyramid data structure using the scale–and–tile approach for supporting interactive access to datasets much larger than main memory.

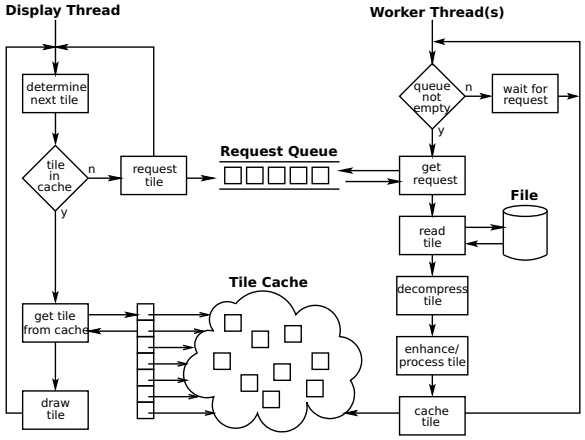

Fig. 2. Multithreading and priority caching system using a tile request queue in KOLAM for parallel reads overlapped with non-blocking interactive visualization.

level (one-half in each dimension) to create a multiresolution structure which is then reorganized into fixed-size tiles [14], [15]. Each tile is individually compressed to create a self-contained subimage. KOLAM can display this tile organization by drawing grid lines to depict the scaled tiles being drawn from the appropriate layer of the pyramid as the user zooms in and out.

In addition to the native pyramid storage format, the open architecture of KOLAM allows for the support of different multiresolution, hierarchical data types. The PSS file format is organized using a scale–and–tile pyramidal structure similar to the native KOLAM datasets and utilize per-tile JPEG compression. Extensions to support JPEG2000 and NCSA HDF5 access are being explored.

In order to provide interactive display updates, KOLAM fetches and displays only those tiles required for the view into the data. In order to display images much larger than primary memory, tiles are cached and paged according to a distance-based metric. The reading and decompression of tiles is accomplished asynchronously using a safe multithreaded architecture. Tile visibility determination for display is managed in a separate thread from those servicing I/O and decompression requests in order to keep the display interactive. A single thread manages the display while multiple read (worker) threads fulfill the tile requests.

KOLAM utilizes a workpile concurrent model by creating

Fig. 3. The KOLAM user interface, showing the various subsystem widgets during a sample session for tracking moving and stationary objects, the lat-long position readout panel in the lower left, filename, ROI and zoom information in the status bar, interactive animation panel, assisted multi-object tracking panel on the right that also controls the display parameters for multiple trajectories and the original and stabilized trajectories trailing behind the two tracked objects.

a queue in which pending operations are enqueued, to be then dequeued by worker threads. KOLAM takes advantage of the manner in which the operating system scheduler manages the blocking and yielding of thread execution for load balancing. The workpile model is used in the KOLAM multithreaded environment by inclusion of the POSIX *pread()* functionality (under Mac OS X and Linux), which ensures that all read operations are atomic (combined seek and read) with respect to the files being handled and ensures that the operations are threadsafe with multiple reader threads. Under the Windows operating system a fully functional *pread()* is not available so KOLAM uses a single reader thread. Figure 2 illustrates the primary display and read/worker threads architecture of KOLAM.

An important theme in the KOLAM GUI design, has been to minimize the number of interaction steps needed for a specific functionality to maximize productivity – such as single-click manual/assisted tracking. The interface components needed for ground truth generation and output visualization from KOLAM-TS are explored below.

A. KOLAM Display Interface

Figure 3 shows the KOLAM user interface along with three subsystem tool interfaces. The KOLAM user interface was designed so that user dialogs can be freely moved and do not occupy a fixed subregion of the main window area. This maximizes the main window region available for motion video visualization. This design choice is useful when multiple monitors are available, in which case the entire display area of the primary monitor can be devoted to visualization, and all dialogs can be positioned on the other display devices in an appropriate user-preferred configuration. Standard functionality such as panning and zooming within large gigapixel imagery is efficiently implemented to provide a smooth interaction user experience. This enables rapid navigation to any region

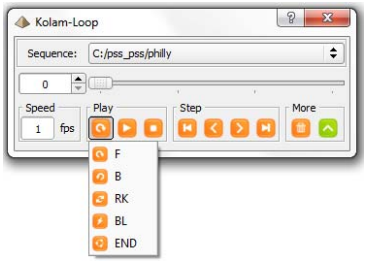

Fig. 4. Animation playback interface in KOLAM supporting forward and backward temporal access, adjustable framerate, random frame access, scrub bar, etc.

of interest at any zoom factor like a virtual light table. An overlay system provides multiple, transparent layers tiered above the actual display area, which serves the dual purpose of presenting additional information superimposed on the display, and enabling marking and annotation of the data. The overlay feature enables the ground truth creation and trajectory display either interactively or using KOLAM-TS.

B. Video Playback/Animation Interface

The KOLAM animation GUI panel in Figure 4 is compact, with an expandable detail tab that provides access to commonly used functionality for seamless playback of multiple WAMI sequences as well as a sequence of (same-sized) image frames in popular formats like TIFF, PNG, JPEG, etc. Bidirectional playback with or without looping is supported, at varying frame rates and stride sizes. It is also possible to single-step forward or backward one frame at a time, or jump to an arbitrary frame in the sequence. Rock (RK) cycles between playing the sequence forward then in reverse order. Blink (BK) animates two adjacent frames that may be separated by an arbitrary stride (step) size. A drop down list shows a selectable playlist of active sequences currently loaded by the user.

C. KOLAM Tracking Interface

The assisted tracking interface in KOLAM is a versatile GUI front-end that interfaces in an open architecture manner to multiple tracking algorithms and allowing tracker specific configuration. These may be grouped as follows: (a) Tracker properties: Creating instances for objects that need to be tracked, selecting a tracker algorithm, switching between automatic (algorithm invocation) and manual (ground truth generation) tracking modes, and invoking the selected tracker on a selected object. (b) Track properties: Setting track visibility, color, and thickness; whether the display dynamically centers itself on the current object, saving to and loading from an archive of previously generated track data, and deleting data for a particular object. Figure 5 shows the KOLAM tracker interface providing real-time feedback via messages whenever any of the above properties are modified, updating the user about tracker initialization, and normal or abnormal tracker termination.

III. KOLAM-TS Middleware

The KOLAM-TrackingSimulator (KOLAM-TS) middleware provides a set of utilities and tools for creating, evaluating and visualizing object trajectories for assisted tracking in WAMI. KOLAM-TS automates the process of tracker algorithm performance testing and benchmarking, as a faster alternative compared to manual target reacquisition by restarting the tracker on the exact frame where it fails, using the ground truth information. The program is also used to determine performance characteristics of a tracker. One key function is to provide the tracker with an accurate template (bounding box) of the object that is being tracked when restarts are needed by taking into account occlusions, similar to how assisted tracking is done.

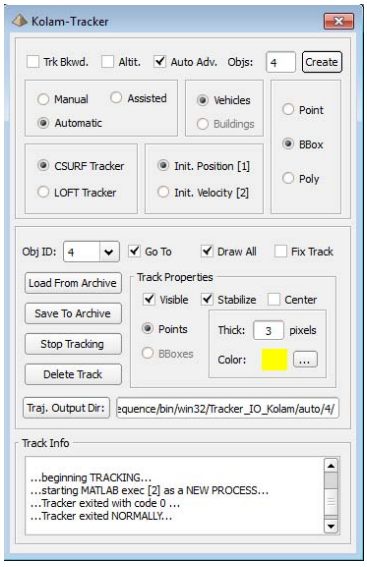

Fig. 5. KOLAM-Tracker interface for specifying tracking parameters: manual vs automatic vs assisted tracking, type of tracker (*i.e.* LOFT), type of trajectories to display (stabilized and unstabilized) and text feedback showing the state of the automatic tracker.

KOLAM-TS uses several different output processing modules for visualization as well as for performance assessment. These modules allow us to combine ground truth centroids and polygons, visibility/occlusion status, and tracker outputs in a unified framework in order to assess tracker behavior post-simulation. This combined information is visualized using KOLAM. The block diagram in Figure 7 describes the overall KOLAM-TS system.

KOLAM-TS has evolved over a period of time to accommodate a variety of trajectory ground truth and WAMI file formats. It currently reads ground truth in XML (structured), KOLAM (flat file), iView (flat file) formats. The initialization of the tracker needs a parameter input file which also acts as one of the inputs to the main harness/simulator module. KOLAM-TS provides flexibility by allowing the user to select partial or complete input from any of the listed sources. The XML structured format is produced by the MIT Layer Annotation Tool [9] as output from ground truth annotation; an example of the MIT LAT GUI is shown in Figure 6. We transitioned from using only centroid information for the target to using both polygon and centroid XML structured tags in order to facilitate automatically restarting trackers during performance testing with accurate, oriented and scaled bounding-box information for the target (vehicle).

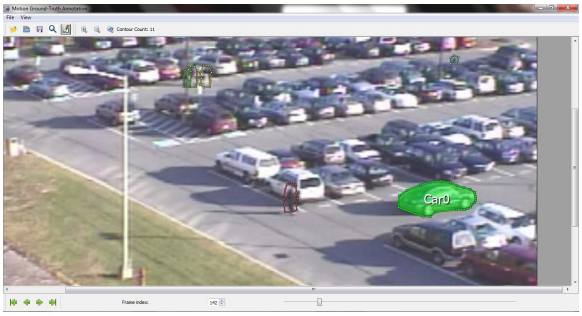

Fig. 6. MIT Layer Annotation Tool [9] for ground truthing showing polygon boundaries around several moving objects and the tracked vehicle highlighted in green.

The MU-Harness component of KOLAM-TS shown in Figure 7, outputs two main types of files – MATLAB files needed for tracker performance evaluation as well as computing bounding box information helpful for static visualization, and KOLAM track files for dynamic trajectory visualization. A high level description of MU-Harness functionality is shown in pseudocode form in Algorithm 1. There are three different parser modules that are embedded within the MU-Harness. The XML parser mainly extracts data from a structured XML file format. It reads the object shape information represented as a polygon and calculates the geometric center of the object (*i.e.* centroid) using Simpson's method. The other two parsers within the MU-Harness are for reading visibility occlusion status and for target initialization (non-oriented bounding box and frame filename) from flat files. In addition to the

parsers embedded within the MU-Harness, we also have five different parsers for conversion between the different file formats shown in purple in Figure 7. This gives us the freedom to collect ground truth using a host of different tools. The XML2KOLAM converter was mainly designed to facilitate the visualization of polygons in KOLAM. The XML2iView and iView2KOLAM essentially perform the same task but they do not include the polygon information while outputting to the KOLAM output trajectory format.

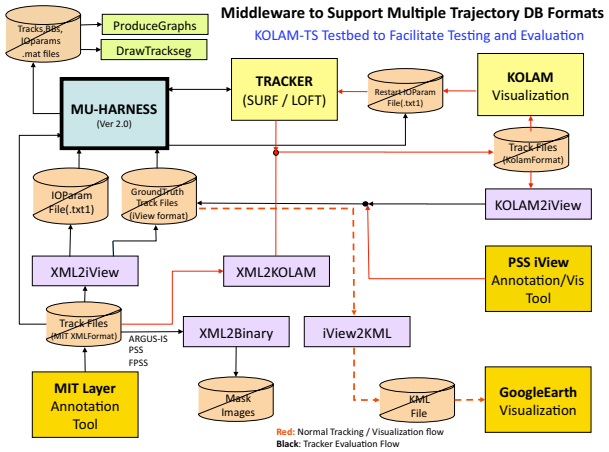

Fig. 7. KOLAM-TS middleware system diagram with external interactions showing the MU-Harness in cyan, track file conversion parsers (MATLAB programs) in purple, the interactive tools in yellow, and the flow of information as directed edges.

Algorithm 1 MU-Harness (MATLAB)

Input: $GTList$ // Centroids, polygons, start & end frame, etc.
Output: $TrackList, BBoxList$

```
//Initialize total number of tracker segments and target object
count ← 0, Target = [ ]
for each (GT ∈ GTList) do
    fr ← GT.start_frame
    while (fr < GT.end_frame) do
        // Update Target information
        Target.start_frame = fr
        Target.start_bbox = GT[fr].bbox
        // Start Tracker
        Track ← Tracker(Target)
        // Tracker returns when (fr > GT.end_frame) .OR.
        // (||Track[fr].xy − GT[fr].xy|| > max_dist)
        count ← count + 1
        TrackList[count] ← Track
        BBoxList[count] ← Target.start_bbox
        fr ← Track.end_frame + 1
        // Re-acquire target only when it is fully visible
        while ( (fr < GT.end_frame) .AND.
        (GT[fr].visible == false) ) do
            fr ← fr + 1
        end while
    end while
end for
```

The need for multiple parsers (eight so far in the KOLAM-TS) points out the benefits of developing a common XML-based Ground-truth Tracking Markup Language (GTML). These parsers were required to convert between and combine different trajectory, meta-data and supporting information file formats. The tools shown in Figure 7 all have their advantages and disadvantages for collecting and managing ground truth information but output metadata in different formats. KOLAM-TS needs to interface with all these formats in order for us to be able to evaluate our tracking algorithm given multiple ground truth sources and dataset formats. The need for a standard format for ground truth has thus become essential in order to effectively share data and results. We propose the use of GTML, as an extension of the XML formats used in ViPER-XML [4] and the MIT Layer Annotation Tool [9]. ViPER and MIT LAT do not support wide-area imagery, ViPER does not provide general polygon shape drawing and MIT LAT does not support marking occlusions. GTML is still in its nascent phase and currently incorporates occlusion status for every frame, bounding polygon information as well as other metadata. The GTML file can further interface with a database for managing a large collection of experiments similar to the ViSOR system [24]. Key-Length-Value (KLV) representation established by the Society of Motion Picture and Television Engineers (SMPTE) is an alternative to XML based metadata that is being adapted by the MISB for WAMI and consists of a 16-byte SMPTE administered universal label (Key), followed by length of the data (Length) then the payload of bytes (Value). KLV coding is very efficient for streaming and high performance applications and is generally more backwards compatible as metadata standards evolve [10].

Figure 8 shows the visualization of tracker produced segments (computed tracks in cyan) along with the ground truth (target tracks) using KOLAM. The yellow trajectory shows the stabilized ground truth or target track and the cyan trajectory segments show the automated tracker output or computed tracks. The marked area shows the location where the tracker failed and was restarted by MU-Harness (Algorithm 1). KOLAM provides dynamic visualization of the trajectory evolution compared to ground truth within the spatial context of occlusions and scene complexity.

IV. Conclusions

The KOLAM environment provides an integrated toolset for: (i) interactive visualization of WAMI datasets (ii) manual ground truth generation of moving and stationary objects (iii) an efficient assisted tracking mode to increase user productivity and (iv) KOLAM-TS to facilitate automated evaluation of tracking algorithms. KOLAM allows the user to interactively step through and view tracked frames, compare trajectories or segments with ground truth. KOLAM-TS uses ground truth information to evaluate the tracker algorithm performance and visualize the track segments, making it convenient to view the image region in the failed frame and/or the restarted frame for debugging and improving the tracker. Together, these tools provide a comprehensive interface for visualization, evaluation and benchmarking of automated object tracking performance versus manual ground truth in wide-area motion imagery.

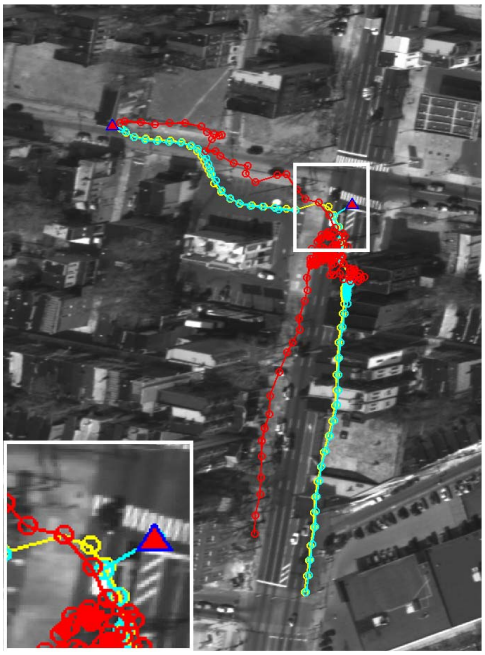

Fig. 8. Visualization of tracking results (Car2, Philly) showing manual ground truth (unstabilized in red and stabilized in yellow) and several stabilized automatic tracker segments (in cyan). One location where MU-Harness restarted the tracker after it failed is shown inside the white box (detailed zoomed view in lower left inset). The red triangles mark the failed frames.

ACKNOWLEDGEMENTS

The PSS database is a WAMI repository collected and maintained by Persistent Surveillance Systems Inc. kindly made available to us by Ross McNutt. Ilker Ersoy provided feedback on improvements to KOLAM and KOLAM-TS and worked on the tracking algorithms. Ariel Abrams-Kundan and Koyeli Ganguli assisted with ground truth creation and parts of KOLAM-TS middleware. This research was partially supported by grants from the U.S. Air Force Research Laboratory (AFRL) under agreement FA8750-11-1-0073 and Leonard Wood Institute (LWI 181223) in cooperation with the U.S. Army Research Laboratory (ARL) under Cooperative Agreement Number W911NF-07-2-0062. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of AFRL, LWI, ARL, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

REFERENCES

[1] Air Force Research Laboratory. Columbus Large Image Format (CLIF) 2007 dataset. *https://www.sdms.afrl.af.mil/datasets/clif2007/.*

[2] C.J. Carrano. Ultra-scale vehicle tracking in low spatial resolution and low frame-rate overhead video. In O.E. Drummon and R. D. Teichgraeber, editors, *SPIE Proc. Signal and Data Processing of Small Targets*, volume 7445, 2009.

[3] N.P. Cuntoor, A. Basharat, A.G.A. Perera, and A. Hoogs. Track initialization in low frame rate and low resolution videos. In *Int. Conf. Pattern Recognition*, pages 3640–3644. IEEE, 2010.

[4] D. Doermann and D. Mihalcik. Tools and techniques for video performance evaluation. In *15th Int. Conf. Pattern Recognition*, volume 4, pages 167–170. http://viper-toolkit.sourceforge.net, 2000.

[5] A. F. Hasler, D. Chesters, M. Jentoft-Nilsen, and K. Palaniappan. High performance animation of GOES weather images. In *SPIE Proc. on GOES-8 and Beyond*, volume 2812, pages 80–83. 1996.

[6] A. F. Hasler, K. Palaniappan, M. Manyin, and J. Dodge. A high performance interactive image spreadsheet (IISS). *Computers in Physics*, 8(4):325–342, 1994.

[7] Home Office Scientific Development Branch. Imagery library for intelligent detection systems i-LIDS. In *IET Conf. Crime and Security*, pages 445–448. http://www.ilids.co.uk/, 2006.

[8] X. Jiangjian, C. Hui, H. Sawhney, and H. Feng. Vehicle detection and tracking in wide field-of-view aerial video. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 679 – 684, 2010.

[9] C. Liu, W.T. Freeman, E.H. Adelson, and Y. Weiss. Human-assisted motion annotation. In *IEEE Conf. Computer Vision and Pattern Recognition*. IEEE, 2008.

[10] NGA Motion Imagery Standards Board. Profile 2: KLV for LVSD Applications (MISB EG 0810.2). http://www.gwg.nga.mil/misb/docs/eg/EG081002.pdf.

[11] A.T. Nghiem, F. Bremond, M. Thonnat, and V. Valentin. ETISEO: Performance evaluation for video surveillance systems. In *IEEE 5th Int. Conf. Advanced Video and Signal Based Surveillance*, 2007.

[12] Object Video. Virtual Video Tool: A Half-Life 2 Mod. *http://development.objectvideo.com/.*

[13] K. Palaniappan, F. Bunyak, P. Kumar, I. Ersoy, S. Jaeger, K. Ganguli, A. Haridas, J. Fraser, R. Rao, and G. Seetharaman. Efficient feature extraction and likelihood fusion for vehicle tracking in low frame rate airborne video. In *13th Int. Conf. Information Fusion*, 2010.

[14] K. Palaniappan and J.B. Fraser. Multiresolution tiling for interactive viewing of large datasets. In *17th Int. Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology*, pages 338–342. American Meteorological Society, 2001.

[15] K. Palaniappan, A.F. Hasler, J.B. Fraser, and M. Manyin. Network-based visualization using the distributed image spreadsheet (DISS). In *17th Int. Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology*, pages 399–403, 2001.

[16] K. Palaniappan, R. Rao, and G. Seetharaman. Wide-area persistent airborne video: Architecture and challenges. In B. Banhu, C. V. Ravishankar, A. K. Roy-Chowdhury, H. Aghajan, and D. Terzopoulos, editors, *Distributed Video Sensor Networks: Research Challenges and Future Directions*, chapter 24, pages 349–371. Springer, 2011.

[17] R. Porter, A.M. Fraser, and D. Hush. Wide-area motion imagery. *IEEE Signal Processing Magazine*, 27(5):56–65, 2010.

[18] F.Z. Qureshi and D. Terzopoulos. Surveillance camera scheduling: A virtual vision approach. *Multimedia Systems*, 12(3):269–283, 2006.

[19] F.Z. Qureshi and D. Terzopoulos. Smart camera networks in virtual reality. *Proceedings IEEE*, 96(10):1640–1656, 2008.

[20] S.K. Ralph, J. Irvine, M.R. Stevens, M. Snorrason, and D. Gwilt. Assessing the performance of an automated video ground truthing application. In *33rd IEEE Workshop Applied Imagery Pattern Recognition*, pages 202–207, 2004.

[21] S. Randall and J. Antonisse. The standard exchange of motion indicators by image-based trackers. In *Proc. SPIE on Geospatial InfoFusion Systems and Solutions for Defense and Security Applications*, volume 8053, 2011.

[22] V. Reilly, H. Idrees, and M. Shah. Detection and tracking of large number of targets in wide area surveillance. In *11th European Conf. Computer Vision*, pages 186–199. Springer-Verlag, 2010.

[23] Jonathan C. Roberts. State of the Art: Coordinated & Multiple Views in Exploratory Visualization. In Gennady Andrienko, Jonathan C. Roberts, and Chris Weaver, editors, *Proc. 5th Int. Conf. Coordinated & Multiple Views in Exploratory Visualization*, July 2007.

[24] R. Vezzani and R. Cucchiara. Annotation collection and online performance evaluation for video surveillance: The ViSOR project. In *IEEE 5th Int. Conf. Advanced Video and Signal Based Surveillance (AVSS)*, pages 227–234, 2008.