J Sign Process Syst
DOI 10.1007/s11265-016-1206-6

A Container-Based Elastic Cloud Architecture for Pseudo Real-Time Exploitation of Wide Area Motion Imagery (WAMI) Stream

Ryan Wu [1] · Bingwei Liu [1] · Yu Chen [1] · Erik Blasch [2] · Haibin Ling [3] · Genshe Chen [4]

Received: 25 June 2015 / Revised: 17 January 2016 / Accepted: 10 November 2016

Abstract Real-time information fusion based on WAMI (Wide-Area Motion Imagery), FMV (Full Motion Video), and text data is highly desired for many mission critical emergency or military applications. However, due to the huge data rate, it is still infeasible to process streaming WAMI in a real-time manner and achieve the goal of online, uninterrupted target tracking. In this paper, a pseudo-real-time Dynamic Data Driven Applications System (DDDAS) WAMI data stream processing scheme is proposed. Taking advantage of the temporal and spatial locality properties, a divide-and-conquer strategy is adopted to overcome the challenge resulting from the large amount of dynamic data. In the *Pseudo Real-time Exploitation of Sub-Area (PRESA) framework*, each WAMI frame is divided into multiple sub-areas and specified sub-areas are assigned to the virtual machines in a container-based cloud computing architecture, which allows dynamic resource provisioning to meet the performance requirements. A prototype has been implemented and the experimental results validate the effectiveness of our approach.

Keywords WAMI (wide-area motion imagery) · Dynamic data-driven application systems · Pseudo-real-time processing · Container-based Cloud

✉ Yu Chen
ychen@binghamton.edu

1 Department of Electrical & Computer Engineering, Binghamton University, SUNY, Binghamton, NY 13902, USA

2 Air Force Research Laboratory, Rome, NY 13440, USA

3 Department of Computer & Information Sciences, Temple University, Philadelphia, PA 19122, USA

4 Intelligent Fusion Technology, Inc., Germantown, MD 20876, USA

1 Introduction

Situational awareness (SAW) is essential for mission critical applications. Object assessment can come from many sources such as cyber, linguistic and surveillance data from which information fusion exploitation techniques are needed [1, 2]. Object detection from surveillance data is often achieved through an exploitation of sensor data such as wide area motion imagery (WAMI) systems in a layered sensor environment [3]. Real time detection is ideal since the faster targets are detected, the faster the opportunities to assess their activities through tracking and identification [4]. However, real-time tracking is difficult due to the complexity of the problem space, cluttered scenes with obscurations, varying sensor resolutions, different environmental conditions (e.g., illumination), and the intelligence of a target. Moreover, WAMI systems typically produce tens of thousands of moving target indicator (MTI) detections for a city-size urban area of only 40 km^2 at video rates of up to 12 Hz [5, 6].

Compared with traditional video surveillance tasks, WAMI surveillance is characterized by its large amount of dynamic data. A typical low frame rate (1.25 Hz) WAMI sequence, generates a data flow of over 100 MB of data per second, or over 400GB per hour. The data scale can be even larger for high frame rate (e.g., >10 Hz) and/or higher resolution videos (e.g., >10 K × 10 K) [7, 8].

With such large data rates, there is a lack of real-time methods to integrate data. The existing methods are static updates at each incident site and therefore the response in such systems is significantly slowed. The ability to integrate real time data to support situational awareness (SAW) target detection would be important [9]. Inherently, effective responses for target detection rely on the level of SAW and data processing, sharing, computation, and analysis [10]. WAMI video data offers tremendous support to target detection in

Published online: 26 November 2016

conjunction with other intelligence data but is very difficult to process and analyze the data due to its size, dynamics, and security requirements [11].

Several methods have been proposed to utilize the Dynamic Data Driven Application System (DDDAS) [12, 13] for target tracking and information fusion [14, 15]. Recent examples demonstrated coordination between UAVs and image sensing [16]. Liu et al. [17] have used the DDDAS concept to combine modeling, measurements, and software solutions for an information fusion method of tracking targets using a cloud architecture [18].

Cloud Computing has been recognized as an ideal candidate that can meet the next-generation large data contextual challenges. However, the current mainstream hypervisor-based Cloud architecture cannot satisfy the requirements of a granular architecture that allows new mission critical applications to be deployed using drastically less computing resources, reducing data management burdens, and maintaining high levels of security. A new solution is needed that dynamically adapts to the changing environment while minimizing the overhead at the service providers' side.

The Cloud Computing using *container-based virtualization* technique does not depend on hypervisor. Instead, the operating system is modified to securely isolate multiple instances of an operating system within a single host machine. The guest operating system instances are often called virtual private servers (VPS), containers, or virtual machines (VMs). Since neither a hypervisor nor privilege instruction trapping/ translation is needed, near-native performance is achieved [19]. Another important feature is that all VMs share a single kernel, which allows for great flexibility through live resource reallocation and ultra-low overhead. Therefore, we adopt a container-based architecture for WAMI processing.

In this paper, we propose a pseudo-real-time WAMI data stream analysis scheme. Taking advantage of the temporal and spatial locality properties, a divide-and-conquer strategy is used to overcome the challenges resulting from the large amount of dynamic data. The WAMI frame is divided into multiple sub-areas, each of which is assigned to a container-based VM. The sub-areas are processed independently of one another and the results are displayed in real-time to an operator. When the operator identifies certain suspicious objects in a sub-area, the resources of the container assigned to it are dynamically allocated to match the performance requirement. Then, the main processing engine keeps fetching new frames, dividing them, and assigning sub-areas to the containers for feature abstracting and target tracking. In this manner, we can process certain "key" areas in real-time even though we still cannot process the entire frame in real-time.

The rest of this paper is organized as follows. Section 2 provides a brief survey on the related work in WAMI processing. Section 3 introduces the rationale of the proposed pseudo-real-time processing approach, and Section 4 presents a system level description of the framework. Section 5 reports the experimental results in detail. Section 6 discusses the design tradeoffs and critical considerations. Section 7 provides conclusions.

2 Related Work

The research in WAMI processing usually focuses on: (1) developing data-driven models to characterize the dynamic objects in the scene, for an excellent overview, see Porter et al. [6]; and (2) improving visual target tracking performance using background registration to compensate the camera motion. Registration can significantly improve the quality of tracking algorithms [20, 21]. However, the registration process requires tremendous computing resources, causing the entire tracking application to only achieve a frame rate of less than one frame per second on a commodity computer. This is obviously unsuitable for real time tracking applications.

Another issue is the large data files for which researchers have developed methods in WAMI compression [22–24]. WAMI processing requires data management [25, 26]. One example is a low-frame evaluation of WAMI tracking and performance assessment as shown by Ling et al. [27]. Numerous methods have been applied to WAMI such as Sparse Representation [28], kernel learning [29], likelihood of features [30], Histogram Based Descriptors [31] to track many targets [32]. K. Liu et al. [33, 34] proposed using optical flow combined with principal component analysis for motion detection. These tracking methods support activity recognition [35], context assessment [36], and enhanced situation awareness [37].

Context is an important element of WAMI analysis including spatial [38] and temporal context [39]. Context provides an assessment of the vehicle directions [40] and support pattern of life analysis [41]. The goal is to maintain Maximum Consistency Context, a spatio-temporal context that is robust to noises in target neighbourhood [42]. Recent efforts have developed methods for testing [43] and real-world assessment [44]. From all of these works, registration, stream processing, and context-based tracking rely on the ability to robustly register the WAMI data in real-time.

The Scale Invariant Feature Transform (SIFT) developed by David Lowe [45] is able to extract invariant features to be used by reliable and robust matching between views of a target or scene. The matching is so robust that it can endure distortion such as scaling, rotation and change of illumination. Using SIFT and Random Sample Consensus (RANSAC) [46] for registration between consecutive frames before applying tracking algorithms, the quality of tracking algorithms can be significantly improved [27].

Although the quality of tracking algorithm can be enhanced by registration, the processing frame rate is very low. Our previous effort of improving the processing speed for target tracking [19, 47] utilized *container based virtualization* to achieve flexible and scalable resource allocation in large scale mission system, which laid a foundation for work reported in this paper. In [19], we focused on serving multiple users using the lightweight virtualization technology of container. The work reported in [47] has improved the performance of a full-motion video (FMV) tracking application by distributing frames to multiple containers. One of the main contributions if this paper is the novel method of dividing frames into subareas and accelerated areas of interests.

3 Rationale of Pseudo-Real-Time Processing

WAMI surveillance systems can monitor expansive and densely populated areas for targets. Modern imaging sensors produce high resolution frames that can capture important details scattered over a large area. Frames are typically passed through a feature detector, which identifies targets that can be tracked in subsequent frames. As established in the previous section, accurate feature detection is fairly computationally expensive on current computer hardware. The high resolution that defines WAMI surveillance prohibits feature detection for all but extremely low frame rate streams.

Real-time stream processing requires that the output frame rate be equal to the stream frame rate. The system must be capable of processing an entire frame before the next frame arrives. While this goal is fully realizable for low resolution streams, attempting to process WAMI streams in real-time consumes more computing resources than available in commodity systems.

It is possible to achieve real-time processing of WAMI surveillance by altering the data stream. In one method, frames can be discarded to allow the image processing system time to keep up with incoming frames. However, this may severely impact the usefulness of detected features since targets would be able to cover larger distances between subsequent frames. With sufficient knowledge of the capabilities of the processing system, targets could subvert tracking efforts and pose undetected threats to the area.

Alternatively, the frame resolution can be reduced to facilitate real-time processing. While this method does not suffer from the tracking problem introduced by frame discarding, it arguably demotes the data stream out of WAMI classification. Capturing small details in the frame, a key benefit of WAMI surveillance, is sacrificed to achieve real-time processing since features may pass undetected or be rendered completely unrecognizable in the low resolution frames.

In addition, both methods suffer a common weakness: dependency on the capabilities of the computing hardware. Sufficiently reducing the stream quality requires intimate knowledge of both the feature detector and the hardware. If either the processing (discarding frames) or hardware (resolution) is altered, then the stream quality must be reconfigured for the new system. This system lacks versatility and is difficult to upgrade.

We propose a *Pseudo Real-time Exploitation of Sub-Area (PRESA) framework* for processing WAMI frames that avoids these pitfalls. Rather than attempt to process the full frame in real-time, our framework instead aims to process a sub-area of the frame in real-time. When an area of interest is identified in the frame, the computing resources are reallocated to accelerate the processing rate of that sub-area to real-time. The resource allocation preserves the detail afforded by WAMI surveillance and meets the time resolution requirement for accurate target tracking within the sub-area. Once the target density in the sub-area returns to normal distribution, the framework redistributes resources across the full frame. The framework takes a dynamic approach to resource allocation to apply computing power to where it is needed in the frame. In this manner, the content of the WAMI stream dictates how the hardware is utilized.

The PRESA framework is built upon *container-based Cloud Computing* platform to achieve the necessary flexibility. The full WAMI frame is divided into a grid of uniformly-sized sub-areas. Each sub-area is assigned to a container for processing. When a sub-area is marked for acceleration to real-time processing, its associated container is allocated a larger share of the computing resources. Even without acceleration, the PRESA framework spatially parallelizes frame processing which may improve feature detector performance, especially for detectors that do not natively support multithreading. Containerization provides a level of abstraction from the hardware that allows the framework to operate independently of the physical computer configuration. Containers can be distributed across multiple nodes in a computing cluster and can migrate between nodes to handle dynamic workloads.

4 Pseudo-Real-Time Processing Framework

The PRESA framework is divided into three distinct groups: clients, workers, and resource managers. *Clients* provide user interfaces to the framework and are designed to run on personal computers. *Workers* perform feature detection on subareas of frames according to client requests and execute exclusively within containers. A single *resource manager* runs on each computing node to allocate resources to containers and receive requests to accelerate containers. Each element of the framework communicates information with the others via Transmission Control Protocol (TCP) socket connections. Figure 1 depicts the high level structure of the PRESA framework for a single client, worker, and resource manager.

When a user wants to process a WAMI stream in the framework, the user opens a *client* and enters the processing

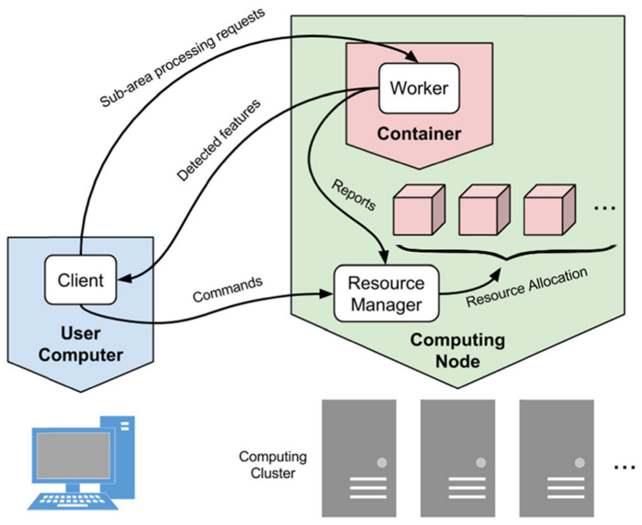


Figure 1 Pseudo-real-time processing framework.

parameters. The user specifies the grid dimensions for dividing the frame and the collection of workers that will be used to process the stream. Once configured, the client assigns a sub-area to each worker and opens a display window for the user. The client starts a thread for each worker to issue job requests and receive detected features. A job request indicates a frame and sub-area for the worker to process.

Workers wait to receive job information from clients over the network. Upon receiving a job request, the worker loads the frame indicated by the request and applies the feature detector to the sub-area. The worker then sends the resulting detected features back to the client according to the network information included in the request. It also calculates the average frame rate from the time interval elapsed while processing the frame and reports this to the resource manager responsible for the computing node.

Once the client receives detected features from a worker, it updates the corresponding sub-area in the user display. The client then prepares the next request for the worker. Since accelerated sub-areas progress through the stream faster than non-accelerated sub-areas, some degree of synchronization is necessary to keep slower sub-areas from lagging behind. If a sub-area is the furthest along in the data stream, then it updates the counter shared by all client threads with its current frame number. If a sub-area is several frames behind the furthest sub-area, then it skips frames to match it. The allowable frame lag is called the frame synchronization tolerance. Unless otherwise noted, a tolerance of zero frame is used to enforce strict synchronization between sub-areas. The synchronization algorithm is shown in Fig. 2.

Additionally, the client is capable of issuing commands to the resource manager to accelerate sub-area processing. When the user clicks on the sub-area of the frame, the client sends a command to the resource manager to allocate more computing resources to the worker assigned to the sub-area. If the sub-area is already accelerated, then the client instead sends a command to restore the default resource allocation.

The *resource manager* allocates computing resources to the containers on its node. The resource manager receives both frame rate reports and acceleration commands from workers and clients, respectively. Frame rate reports provide feedback for the proportional controller that determines the computing resources distribution between accelerated containers. A simple indicator can be the processing frame rate of each individual container. Acceleration commands set the target frame rate for faster processing containers. The resource manager only updates the resource distribution when a report or command is received which reduces the impact on node performance when waiting for a message type.

Once the last frame of the WAMI stream is processed, the client issues commands to each of the resource managers to restore the default resource allocation to all containers. This releases the workers for other clients to use.

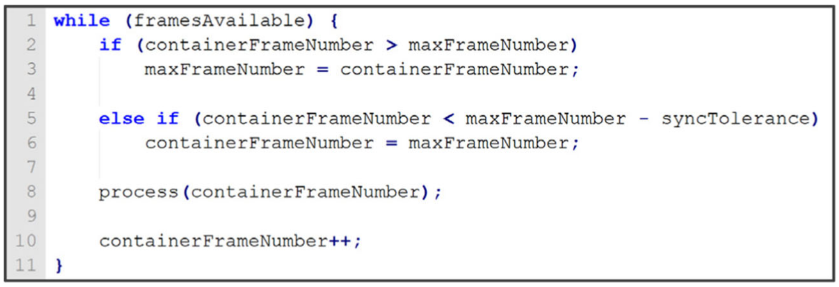
5 Experimental Validation

5.1 Experimental Setup

Our experiments were conducted on a container-based cloud computing platform at Binghamton University. The platform was comprised of four identical computing nodes. Each node had a Xeon E5–2509 quad core CPU at 2.4GHz, 16GB memory and 3 TB storage. The nodes were installed with CentOS release 6.4. The kernel was patched with OpenVZ [48] 2.6.32-042stab085.17. The containers ran Ubuntu 14.04.1 LTS and were installed with OpenCV 2.4.8, which provided the SIFT feature detector and client display interface, and Boost 1.54.0, which provided multithreading support.

We tested the framework using the SIFT feature detector with WAMI data stream composed of 171 frames from AFRL CLIF 2006 dataset [49]. The data stream captures a large area from an aerial viewpoint and represents the typical stream processed by the framework. Each frame is a 2672 by 1200 pixel grayscale image stored in JPEG format. Figure 3(a) is an example WAMI aerial picture and Fig. 3(b) shows the SIFT detector key-points on the picture. Figure 3(c) shows an output frame from the client program with accelerated sub-areas highlighted in red. The user display was disabled when collecting frame rate data since refreshing the display in OpenCV constrained the output frame rate to a maximum of 4 frames per second (FPS). The framework was capable of processing the WAMI stream at higher frame rates than that, especially when employing container acceleration. Similarly, we assumed that a copy of the data stream is stored locally in each container to isolate the effect of the framework on computing resource utilization. Determining the optimal method

Figure 2 Frame synchronization algorithm.



```
while (framesAvailable) {
    if (containerFrameNumber > maxFrameNumber)
        maxFrameNumber = containerFrameNumber;

    else if (containerFrameNumber < maxFrameNumber - syncTolerance)
        containerFrameNumber = maxFrameNumber;

    process(containerFrameNumber);

    containerFrameNumber++;
}
```

for displaying and moving WAMI data streams across a network is beyond the scope of this study.

5.2 Case 1: Single Node Experimental Results

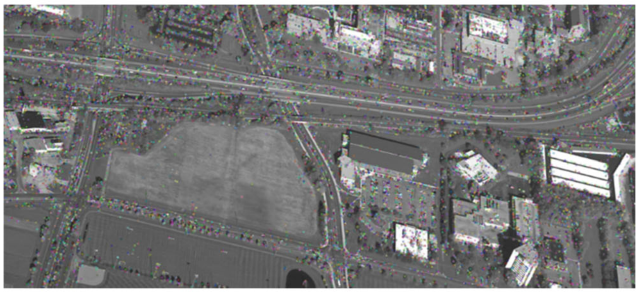
We first tested the PRESA framework with a naïve case in which a single worker processes the WAMI stream. This provides a baseline for comparing the effects of frame division and redistributing resources. The client loaded the data stream and assigned the worker to apply the SIFT feature detector to the full frame. The resource manager on the node collected reports from the worker, but the resource allocation was not changed for the duration of the test.

The worker's output frame rate was sampled at 100 millisecond intervals. The samples were passed through averaging filter with a window of 1.5 s to smooth out small differences between adjacent samples. The frame rate samples for a single pass through the data stream are plotted against time in Fig. 4. The single worker configuration required more than 300 s (5 min) to process the data stream in full frames. The worker maintained a steady output of approximately 0.5 FPS for the duration of the test.

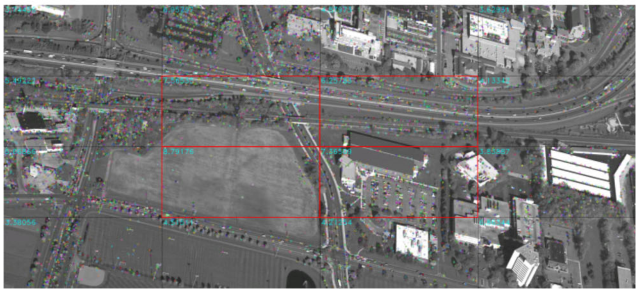
Next, we tested the effect of distributing processing work across multiple containers on a single node. As described in Section 3, the framework is capable of dividing frames into sub-areas for processing by different containers. Spatially dividing frames parallelizes the computationally intensive feature detection operation. The client divided the full frame along a grid into a number of sub-areas, each of which was assigned to a container for processing. Since all containers were located on a single node for this test, only a single resource manager was needed to collect reports. The resource manager did not redistribute resources for the test duration.

In this manner, the framework was tested for varying degrees of frame division. In the most extreme case, the full frame was divided into an 8 by 8 grid, distributing processing work

(a) **A WAMI Aerial Picture.**



(b) **SIFT Detector Key-Pooints on WAMI Picture.**



(c) **Spatial Frame Division on WAMI Picture.**

Figure 3 WAMI aerial picture sample.

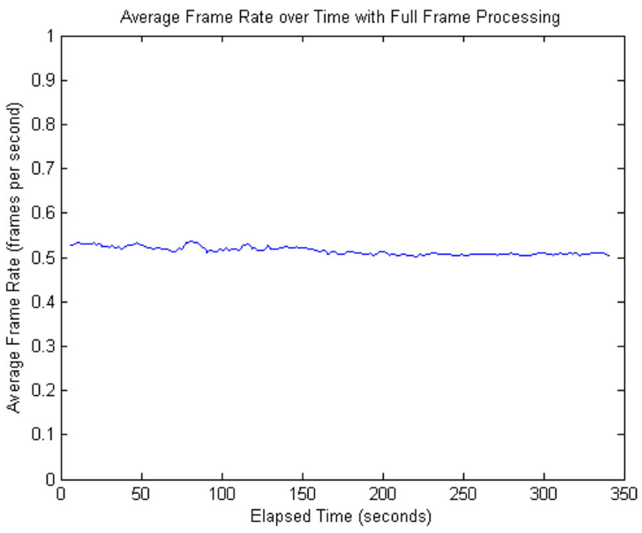


Figure 4 Full frame processing on a single container.

across 64 containers on a single node. For each trial, the WAMI stream was processed by the framework and the total elapsed time was recorded. The average worker output frame rate was calculated by dividing the elapsed time by the number of frames in the data stream. The average frame rates are plotted against the number of containers in Fig. 5. The average frame rate increased linearly at a rate of approximately 0.5 FPS per container. This trend continues until eight or more containers were used to process frames. The frame rate continued to increase to a maximum of 5 FPS for 16 containers. After this turning point, the average frame rate decreased at a steady rate.

We then tested the framework's ability to accelerate feature detection in sub-areas of the frame. After dividing frames into sub-areas, the client issued a command to the resource manager to accelerate feature detection in a sub-area. The resource manager then allocated additional computing resources to the container responsible for the sub-area.

In this test, the client divided the frame into 16 sub-areas, each of which was assigned to a different container. As in the previous setups, all containers ran on the same node so only a single resource manager was required. The client issued an acceleration command for four sub-areas at the 15-s mark. In response, the resource manager allocated more computing resources to the corresponding containers. At the 30-s mark, the client issued a deceleration command for each of the accelerated sub-areas. The resource manager restored the default resource allocation to the containers for the remainder of the test. The output frame rate for each worker was recorded at 100 millisecond intervals and passed through an averaging filter with a 1.5 s window. The average frame rate is plotted at each instance of time for accelerated and non-accelerated sub-areas in Fig. 6.

Before any sub-areas were accelerated, the average output frame rate was approximately 4 FPS. Once the acceleration commands were issued at the 15-s mark, the frame rates of the two types of sub-frames clearly diverged. The average frame rate of the accelerated sub-area increased rapidly to above 6 FPS. By comparison, the average frame rate of the non-accelerated sub-area decreased to approximately 3.5 FPS. This is because more resources are allocated to the accelerated sub-areas than the non-accelerated ones given the same total amount of CPU cycles. Deceleration commands were issued at the 30-s mark, when the frame rates for both sub-areas quickly go back to the same level as before the acceleration command has been issued.

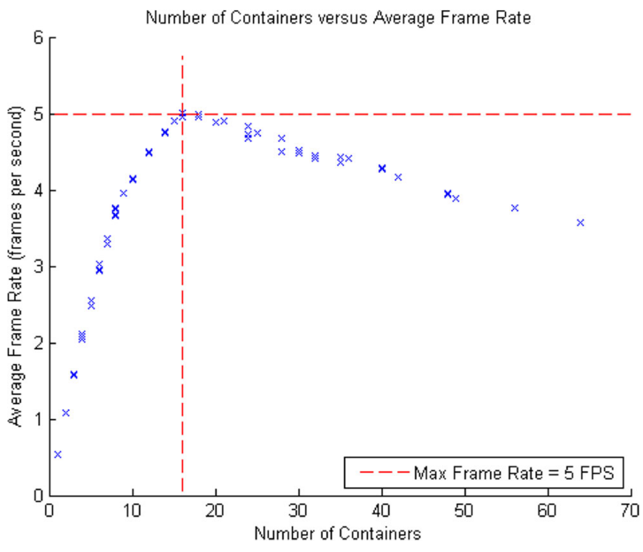


Figure 5 Single node average frame rates.

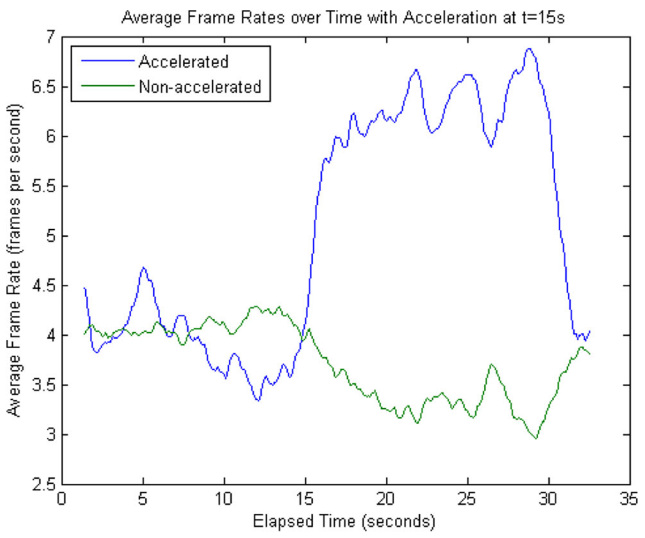


Figure 6 Accelerated versus non-accelerated containers.

5.3 Case 2: Multi-Node Experimental Results

The PRESA framework is capable of coordinating work between containers across multiple hardware nodes. We tested this capability on a cluster of four identical computing nodes. Each computing node had a resource manager, but the container resources were not altered for the duration of the test. A single client assigned sub-frames to nodes in a simple round-robin scheduling pattern to evenly distribute processing work between available nodes. Each node had a number of containers already running that the client can request for processing sub-frames of the WAMI stream.

The test was conducted with two, three and four nodes available to the framework. The number of containers involved in each setup varies from one up to 64 containers. The frame rates are plotted against the number of containers for each setup in Fig. 7. The results for the single node setup are included for comparison.

The framework performance can be divided in two distinct sections: a region of linear increase in frame rate per additional container followed by a region of diminishing slope for greater numbers of containers. The linear region was observed in the single node results earlier, but the trend was even clearer with the additional data from the multi-node setups. The

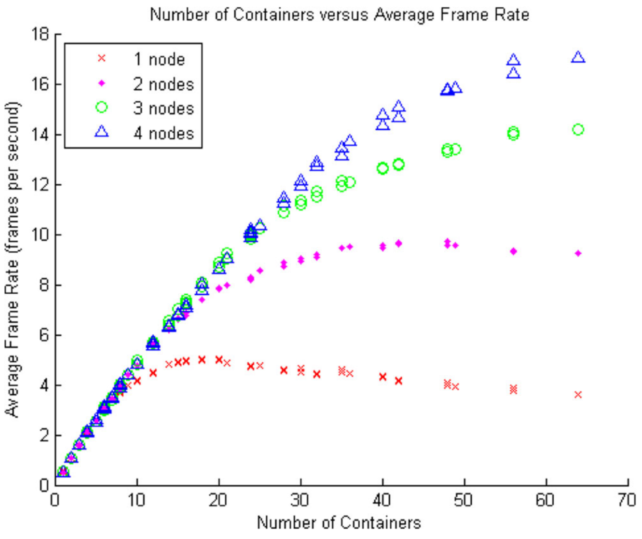


Figure 7 Average frame rate for multiple nodes.

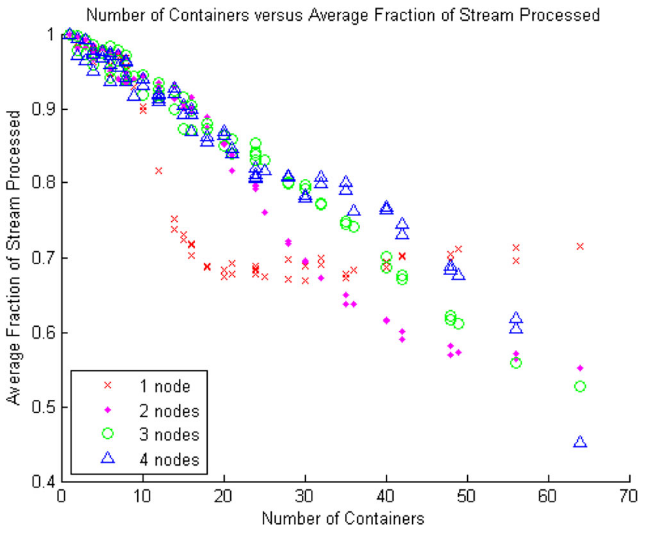


Figure 8 Average fraction of stream processed.

average frame rate increased at approximately 0.5 FPS per additional container in the setup. This relationship was held in each of the configurations. Increasing the number of nodes in the framework extended the width of the linear region by approximately 8 containers per additional node.

In the second region the frame rate rose nonlinearly as the number of containers was increased. The rate at which the frame rate rose starts to diminish as more containers were added to the framework. This diminishing effect was less pronounced as the number of nodes increases, resulting in a gradual slope change for the 4 node setup. Only the single and 2 node setups had frame rates that clearly decreased when the number of containers was set significantly beyond the linear region.

We investigated the mechanism by which the PRESA framework continued to achieve higher frame rates in the second region. Viewing the user display from the client program verified that the framework produces a continuous stream of output frames. However, some sub-areas visibly slowed down relative to other sub-areas due to uneven CPU scheduling to containers. This meant that the frame synchronization mechanism, which is described in Section 4, was skipping frames in slower sub-areas to keep them apace with the rest of the frame. Even though frame synchronization is necessary to produce a useful output frame, excessive skipping reduces the total computing work for the stream. This effect accounts for the observed increase in frame rate while the actual amount of work done remains unchanged.

We quantified the magnitude of frame synchronization by determining the fraction of the stream that is actually processed. The synchronization was calculated from the number of sub-areas skipped for a pass through the WAMI stream. The results are plotted for each of the node setups in Fig. 8. Almost every frame of the stream was fully processed when only a few containers were used, implying that containers received equal and frequent access to computing resources. Increasing the number of containers decreased the fraction of the stream processed since more frame skipping occurred. However, the precise relationship between the two quantities differed depending on the number of nodes.

5.4 Effective Frame Rate (EFR)

We propose a parameter called *effective frame rate* (EFR) that takes the fraction of the stream processed in account. This parameter considers the recorded average frame rate as shown in Fig. 7 along with the fraction of the stream processed as shown in Fig. 8. Specifically, the EFR is defined as the product of the two quantities.

$$EFR = \text{Frame} \quad \text{Rate} \times \text{Fraction} \quad \text{Processed}$$

We tested the EFR parameter with variant frame synchronization tolerances. The framework was given a single node with 16 containers. The WAMI stream was processed with the frame synchronization tolerance ranging from the default zero frame up to 15 frames. The recorded average frame rate and computed EFR are plotted against the tolerance in Fig. 9. The recorded frame rate decreased as the frame synchronization tolerance increased because the frequency at which the synchronization mechanism was triggered decreased. However, the EFR remained at a nearly constant 3.5 FPS. In fact, the recorded average frame rate approached the EFR for greater frame tolerances. Results confirmed that the EFR is invariant to the synchronization mechanism and thus is more appropriate for comparing the framework performance to other methods for utilizing multiple computing nodes.

Figure 10 shows the EFR computed from the results in Fig. 7. The linear region was mostly unchanged since the fraction of stream processed is fairly close to *1* as shown in Fig. 8. The nonlinear region was much less gradual and flattens out almost immediately. In fact, the EFR decreased

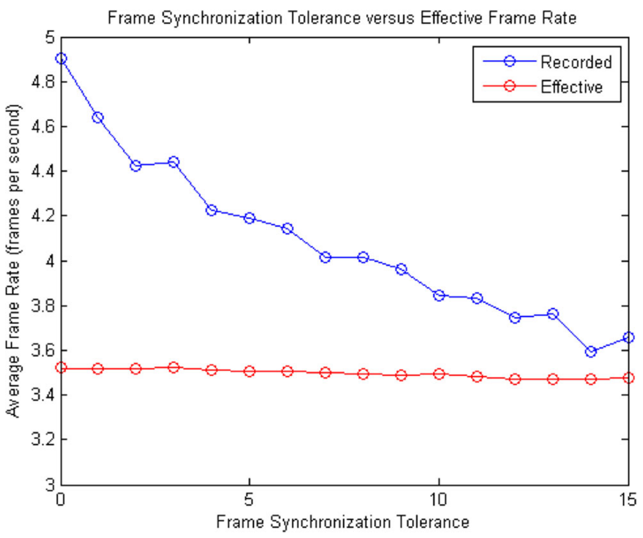


Figure 9 Effective frame rate (EFR) versus synchronization.

slightly as the number of containers was significantly increased as expected. Unlike the average frame rate, EFR provides a clear distinction between the two regions at which the maximum EFR occurs. Increasing the number of nodes increased the number of containers required to reach the maximum EFR by nearly a constant amount.

The results for multiple nodes are summarized in Table 1. The maximum average frame rate and EFR are shown for each of the multiple node setups.

6 Discussions

The *Pseudo Real-time Exploitation of Sub-Area (PRESA)* framework possesses several advantages over applying the feature detector to each full frame. Next,

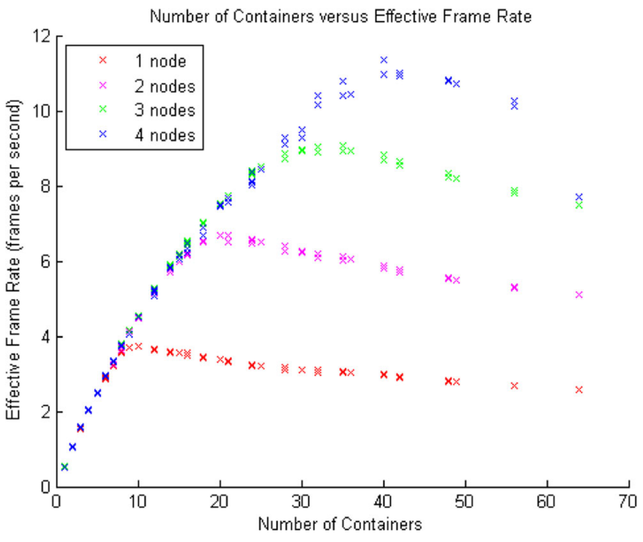


Figure 10 Effective frame rates for multiple nodes.

Table 1 Multi-node results summary.

Nodes	Average frame rate (fps)	Effective frame rate (fps)
1	5.05	3.74
2	9.71	6.70
3	14.20	9.06
4	17.03	11.35

we discuss the major advantages of our PRESA scheme and analyze the impact on the feature detection quality. An experimental study shows that PRESA outperforms the Hadoop-based solution.

6.1 Major Advantages

Frame division provides *significance performance gains for feature detection*. The EFR for the single node setup reached a maximum 3.8 FPS when eight containers were assigned to the framework. The number of sub-areas matches the maximum number of concurrently running threads on the node. This can be generalized to imply that the optimal number of containers for utilizing computing resources is equal to the number of concurrent execution threads supported by the hardware. Subdividing a task between multiple containers is akin to parallelizing work via multithreading. However, containers have the added benefit of enabling processing collaboration and migration between hardware nodes.

Sub-area acceleration allows the framework to achieve even *higher average frame rates* than frame division alone. For tests involving 16 sub-areas, accelerated sub-areas were capable of maintaining a steady output frame rate over 6 FPS, which was significantly higher than the average frame rate of 4 FPS without redistributing resources. The results indicate that the higher accelerated frame rates were possible because computing resources were taken away from non-accelerated sub-areas, which produced lower frame rates than normal. The framework allows for efficient utilization of limited computing resources by applying more processing time to high-interest areas.

The PRESA framework is *effective when extended to span multi-node setups*. The maximum EFR increased by almost a factor of 2 with the first additional node. The rate of increase diminished slowly as more nodes are added to the framework. This demonstrates that the framework utilizes almost the full processing capabilities of the hardware nodes, but the increased overhead from inter-node network delay becomes appreciable with more nodes. The EFR reached a well-defined peak between the linear region of increase and subsequent region of decrease. For the tested computing cluster, the single node reached the peak EFR with 10 containers. Each additional node increased the necessary number of containers by another 10 containers.

Using the PRESA framework, *the peak framework performance can be achieved for a computing cluster of arbitrary size* by experimentally determining the number of containers needed to maximize the EFR for a single node and multiplying that quantity by the total number of nodes in the cluster. Additional tests on different hardware setups are needed to verify this hypothesis.

Increasing the number of sub-areas beyond the degree of concurrency allowed by the processor generally resulted in a gradual reduction in EFR. This implies that the number of containers can be larger than the optimal point without incurring significant overhead. The exception to this trend is the 4 node setup for which the EFR decreased more rapidly after the turning point; further investigation with more nodes may indicate which of the two trends dominates.

6.2 Feature Detection Quality

An aspect of the framework that must be considered is the effect it has on feature detection quality. The performance gains provided by the framework are only useful if the detected features are usable for tracking targets. To assess feature quality for tracking, we compared the accuracy of detected features between the framework and the standalone feature detector.

We generated a list of baseline features by applying the SIFT feature detector to a frame. The same frame was divided into varying numbers of sub-areas before applying the SIFT feature detector. The resulting features were compared with the baseline features to rate the accuracy on a scale from *0* (no matching features) and *1* (identical features). Features were considered to match if their coordinates were equivalent when rounded to the nearest whole pixel. Features that did not match, either missed features not detected by the framework or false features not in the reference list, were counted as errors. The accuracy was computed by taking the difference between number of baseline features and the number of errors and dividing it by the number of baseline features.

$$\text{Accuracy} = \frac{|\text{baseline}| - |\text{Error}|}{|\text{baseline}|}$$

Figure 11 shows the framework accuracy when applying the SIFT feature detector against the number of sub-areas that the frame is divided into, ranging from grids of 1 by 1 to 10 by 10. The accuracy generally decreased as the degree of frame division increased. The rate of decrease diminished as the frame was divided into more sub-areas. The accuracy remains strictly above 0.9 when the frame was divided into less than 10 sub-areas, but numerous configurations of rows and columns achieved similar accuracy for up to 40–50 sub-areas. Figure 11 implies that the SIFT feature detector accuracy is affected by not only the degree of frame division but also the grid dimensions.

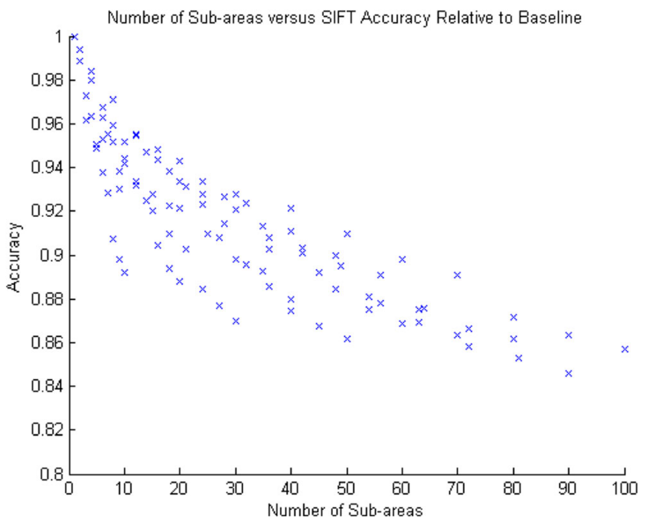


Figure 11 SIFT accuracy relative to baseline.

Even so, the relationship between accuracy and degree of frame division observed with the SIFT feature detector cannot be immediately generalized to cover other feature detectors. Additional investigation is required to determine how different feature detectors are affected by spatial frame division, both in terms of accuracy and performance gains.

6.3 A Comparison Study with Hadoop

It is important to consider how the PRESA framework compares to existing technologies for utilizing computing clusters. One such alternative for improving the performance of WAMI processing algorithms is to simultaneously run many instances of the algorithm on different sets of the image sequence on a distributed system like Hadoop.

To test the performance of SIFT detector, an OpenCV image processing system on Hadoop was built. The SIFT detector algorithm was modified to work with Hadoop framework. The Hadoop YARN framework is designed to run each worker on a resource container that is similar to OpenVZ container. Each mapper process one frame to and output all SIFT key points. The experiment tracks the frame rate for different number of mappers (containers) in Hadoop. Figure 12 compares the output frame rates between PRESA and Hadoop.

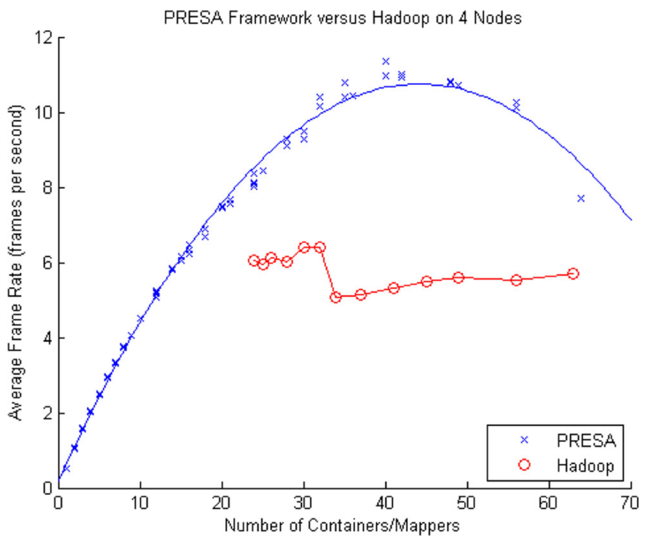


Figure 12 PRESA framework and hadoop comparison.

When provided with equivalent computing hardware, the *PRESA framework manages to exceed Hadoop* in terms of output frame rate. The framework EFR was significantly higher than the output frame rate from the Hadoop tests. However, the framework behaves quite differently when additional containers were added compared to when additional mappers were added to Hadoop. The framework EFR followed a distinct concave down curve while the Hadoop output frame rate was relatively flat. The results demonstrated that the divide-and-conquer method can achieve better performance compared to simply parallelizing by frames. Additional metrics may provide better insight into the fundamental differences between the two methodologies.

7 Conclusions

This paper presents a novel divide-and-conquer strategy that enables us to track suspicious targets in huge WAMI data streams in a pseudo-real-time manner. By identifying and assigning certain sub-areas of a WAMI picture to a container-based virtual machine (VM), in which resources are managed elastically corresponding to the computing task requirements. The proposed Cloud Computing platform can accelerate the processing speed of the areas where suspicious objects are allocated. Through intensive experiments, we have verified the effectiveness of the proposed scheme.

While our work has conceptually validated the pseudo-real-time WAMI process, there are still several challenging issues to be solved before the proposed scheme can be adopted in real-world applications. The team is exploring solutions from different aspects: 1) investigating more flexible and adaptive frame division methods to minimize impact on feature detection quality; 2) adapting architecture level optimization that boosts the processing speed; 3) designing human-machine interface that allows human rationale be integrated into this dynamic tracking framework; 4) exploring whether individual processing of each container can provide important information to the client to help decision making; and 5) investigating the potential optimal number of frame sections that can be accelerated given a certain number of nodes and containers.

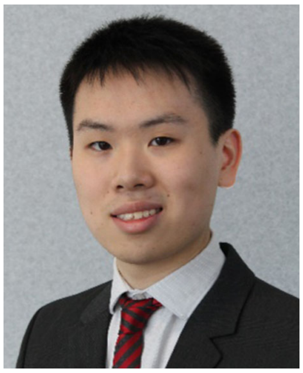
Acknowledgements This work is supported by the US Air Force Research Laboratory (AFRL) Visiting Faculty Research Program (VFRP) and the grant from AFOSR in Dynamic Data-Driven Application Systems. Ryan Wu was a summer undergraduate AFRL research fellow.

The authors also want to express our gratitude to Dr. Erkang Cheng for his valuable suggestions and discussions on SIFT data set and algorithms.

References

1. Blasch E., Bosse E., Lambert D. A. (2012). High-level information fusion management and systems design, Artech House, Norwood, MA.
2. Blasch E., Steinberg A., Das S., Llinas J., Chong C.-Y., Kessler O., Waltz E., White F. (2013). Revisiting the JDL model for information exploitation, Int'l Conf. on Info Fusion.
3. O. Mendoza-Schrock, Patrick, J. A., et al. (2009). Video image registration evaluation for a layered sensing environment, Proc. IEEE Nat. Aerospace Electronics Conf. (NAECON).
4. Blasch E., Yang C., Kadar I. (2014). Summary of tracking and identification methods, Proc. SPIE, Vol. 9119.
5. Porter R., Ruggiero C., Morrison J. D. (2009). A framework for activity detection in wide-area motion imagery, Proc. SPIE, Vol. 7341.
6. Porter, R., Fraser, A. M., & Hush, D. (September 2010). Wide-area motion imagery: narrowing the semantic gap. *IEEE Signal Processing Magazine, 27*(5), 56–65.
7. Blasch E., Seetharaman G., Suddarth S., Palaniappan K., Chen G., Ling H., Basharat A. (2014). Summary of methods in wide-area motion imagery (WAMI), Proc. SPIE, Vol. 9089.
8. Asari V. K. (ed.) (2014). Wide area surveillance: real-time motion detection systems, section augmented vision and reality, Vol. 6, Springer. http://link.springer.com/book/10.1007%2F978-3-642-37841-6.
9. Kahler B. and Blasch E. (2008). Sensor management fusion using operating conditions, Proc. IEEE Nat. Aerospace Electronics Conf (NAECON).
10. Blasch E., Steinberg A., Das S., Llinas J., Chong C.-Y., Kessler O., Waltz E., and White F. (2013). Revisiting the JDL model for information exploitation, Int'l Conf. on Info Fusion.
11. Sun Z. H., Leotta M., Hoogs A. J., Blue R., et al. (2014). Vehicle change detection from aerial imagery using detection response maps, Proc. SPIE, Vol. 9089.
12. Darema F. (2000). DDDAS workshop groups. Creating a dynamic and symbiotic coupling of application/simulations with measurements/experiments. NSF DDDAS 2000 Workshop. Available via www.1dddas.org [accessed Jan 2015].
13. Darema, F. (2005). Grid computing and beyond: the context of dynamic data driven applications systems. *Proceedings IEEE, 93*(3), 692–697.

14. Blasch, E., Seetharaman, G., & Reinhardt, K. (2013). Dynamic data driven applications system concept for information fusion. *Procedia Computer Science, 18*, 1999–2007.
15. Blasch E., Seetharaman G., Darema F. (2013). Dynamic data driven applications systems (dddas) modeling for automatic target recognition, Proc. SPIE, Vol. 8744.
16. Ravela, S. (2012). Quantifying uncertainty for coherent structures. *Procedia Computer Science, 9*, 1187–1196.
17. Liu, B., Blasch, E., Chen, Y., Hadiks, A., Shen, D., Chen, G., & Aved, A. J. (2014). Information fusion in a cloud computing era: a systems-level perspective. *Aerospace and Electronic Systems Magazine, IEEE, 19*(10), 16–24.
18. Liu K., Liu B., Blasch E., Shen D., Wang Z., Ling H., Chen G. (2015). A cloud infrastructure for target detection and tracking using audio and video fusion, IEEE CVPR Workshop.
19. Wu R., Chen Y., Blasch E., Liu B., Chen G., and Shen D. (2014). A container-based elastic cloud architecture for real-time full-motion video (FMV) target tracking, Applied Imagery Pattern Recognition Workshop (AIPR) IEEE, vol., no., pp. 1,8, 14–16 Oct. 2014
20. Pritt M. D., LaTourette K. J. (2011). Automated georegistration of motion imagery, Applied Imagery Pattern Recognition.
21. Wu Y., Chen G., Blasch E., Ling H. (2012). Feature based background registration in wide area motion imagery, Proc. SPIE, Vol. 8402.
22. Palaniappan K., Bunyak F., Kumar P., Ersoy I., Jeager S., Ganguli K., Haridas A., Fraser J., Rao R. M., and Seetharaman G., (2010). Efficient feature extraction and likelihood fusion for vehicle tracking in low frame rate airborne video, Intl. Conf. on Information Fusion.
23. Perera, A. G. A., Collins, R., & Hoods, A. (2008). *Evaluation of compression schemes for wide area video*. IEEE Applied Imagery Pattern Recognition Workshop.
24. Irvine J. M., Israel S. A. (2012). Quantifying interpretability loss due to image compression, Ch. 3 in Video Compression, A. Punchihewa (Ed.), InTech.
25. Blasch E., Seetharaman G., Russell S. (2011).Wide-Area Video Exploitation (WAVE) Joint data management (JDM) for layered sensing, Proc. SPIE, Vol. 8050.
26. Blasch E., Russell S., Seetharaman G. (2011). Joint data management for MOVINT data-to-decision making, Int. Conf. on Info Fusion.
27. Ling H., Wu Y., Blasch E., Chen G., Bai L. (2011). Evaluation of visual tracking in extremely low frame rate wide area motion imagery, Int. Conf. on Info Fusion.
28. Wu Y., Ling H., Blasch E., Chen G., Bai L. (2011) .Visual tracking based on log-Euclidean Riemannian sparse representation, Int. Symp. on Adv. in Visual Computing - Lecture Notes in Computer Science.
29. Liang P., Teodoro G., Ling H., Blasch E., Chen G., Bai L. (2012). Multiple kernel learning for vehicle detection in wide area motion imagery, Int. Conf. on Info Fusion.
30. Palaniappan E K., Seetharaman G., Rao R. M. (2012). Interactive target tracking for persistent wide-area surveillance, Proc. SPIE, Vol. 8396.
31. Mathew, V., & Asari, K. (2012). Local histogram based descriptor for tracking in wide area imagery. *Wireless Networks and Computational Intelligence Comm. in Computer and Information Science, 292*(2012), 119–128.
32. Prokaj J., Zhao X., Medioni G. (2012). Tracking many vehicles in wide area aerial surveillance, IEEE Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW).
33. Liu, K., Du, Q., Yang, H., & Ma, B. (2010). Optical flow and principal component analysis-based motion detection in outdoor videos. *EURASIP Journal on Advances in Signal Processing, 2010*, 1.
34. Liu K., Yang H., Ma B., and Du Q. (2010). A joint optical flow and principal component analysis approach for motion detection. In Acoustics Speech and Signal Processing (ICASSP), 2010 I.E. International Conference on, pp. 1178–1181. IEEE.
35. Choi J., Dumortier Y., Prokaj J., Medioni, G. (2012). Activity recognition in wide aerial video surveillance using entity relationship models, 2012. In International Conference on Advances in GIS, SIGSPATIAL, pages 466–469.
36. Shi X., Ling H., Blasch E., Hu W. (2012). Context-driven moving vehicle detection in wide area motion imagery, Int'l Conf. on Pattern Recognition (ICPR).
37. Blasch E., Seetharaman G., Palaniappan K., Ling H., Chen G. (2012). Wide-area motion imagery (WAMI) exploitation tools for enhanced situation awareness, IEEE Applied Imagery Pattern Recognition Workshop.
38. Liang P., Shen D., Blasch E., Pham K., Wang Z., Chen G., Ling H. (2013). Spatial context for moving vehicle detection in wide area motion imagery with multiple kernel learning. Proc. SPIE, Vol. 8751.
39. Liang P., Ling H., Blasch E., Seetharaman G., Shen D., Chen G. (2013). Vehicle detection in wide area aerial surveillance using temporal context, Int'l Conf. on Info Fusion.
40. Santhaseelan V., Asari V. K. (2013). Tracking in wide area motion imagery using phase vector fields, IEEE Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW).
41. Gao J., Ling H., Blasch E., Pham K., Wang Z., Chen G. (2013) Pattern of life from WAMI objects tracking based on visual context-aware tracking and infusion network models**,** Proc. SPIE, Vol. 8745.
42. Shi X., Li P., Hu W. et al., (2013). Using maximum consistency context for multiple target Association in Wide Area Traffic Scenes, Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP).
43. Pang Y., Shen D., Chen G., Liang P., et al. (2013). Low frame rate video target localization and tracking testbed, Proc. SPIE, Vol. 8742.
44. Basharat, A, Turek, M., Xu, Y., Atkins, C., Stoup, D., Fieldhouse, K., Tunison, P., Hoogs, A. (2014). Real-time multi-target tracking at 210 megapixels/second in wide area motion imagery, IEEE Winter Conf. on Apps. of Computer Vision (WACV).
45. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*(2), 91–110.
46. Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM, 24*(6), 381–395.
47. B. Liu, Y. Chen, D. Shen, G. Chen, K. Pham, E. Blasch, and B. Rubin, "An adaptive process-based cloud infrastructure for space situational awareness applications," in Proc. SPIE, vol. 9085, 2014.
48. SWSoft, "Openvz server virtualization," http://www.openvz.org/, 2006.
49. Columbus Large Image Format (CLIF). (2006). Dataset. https://www.sdms.afrl.af.mil/index.php? collection = clif2006.
50. Hytla P. C., Jackovitz K.S., Balster E.J., Vasquez J. R., Talbert M. L. (2012) Detection and tracking performance with compressed wide area motion imagery, IEEE Nat. Aerospace and Electronics Conference.
51. Liu, K., Ma, B., Du, Q., & Chen, G. (2012). Fast motion detection from airborne videos using graphics processing unit. *Journal of Applied Remote Sensing, 6*(1), 061505–061501.



Mr. Ryan Wu received a B.S. in computer engineering from Binghamton University in 2015. In 2014 he interned at the Air Force Research Lab (AFRL) in Rome, NY researching dynamic reprovisioning capabilities of container-based virtualization and its applications to distributed computing. He presented this research at the Applied Imagery Pattern Recognition Workshop in 2014, for which he received the Best Student Poster Award. Ryan joined Siege Technologies, LLC in 2015 as a computer scientist. Currently, he develops Siege's thin ARM hypervisor for advanced security research on mobile platforms.

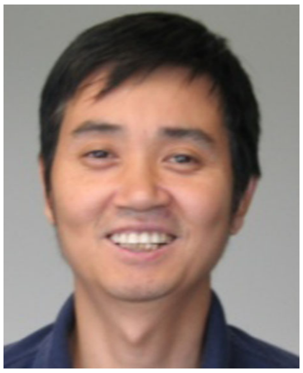
Dr. Bingwei Liu is a Data Science Analyst at Aetna Inc., USA. He received his Ph.D. degree in Electrical and Computer Engineering from the State University of New York (SUNY) at Binghamton in 2015. He received the M.S. in Computer Software and Theory and the B.S. degree in Mathematics from Beijing Normal University, Beijing, China. His current research interests include machine learning for large scale data sets, big data architecture, hybrid Cloud, stochastic models for large scale Cloud data centers, mobile Cloud for telemedicine, Cloud security and target tracking using Cloud. He has over 20 published academic journal, conference papers, book chapter on Cloud Computing and Big data. He also holds one US Patent. Dr. Liu is frequently invited to review scientific papers for leading journals and conferences, and has reviewed over 32 papers as of October 2016. He is an active member of IEEE Cloud Computing Community. He is serving as the track co-chair for the Cloud Computing track of the International Conference on Consumer Electronics 2017 and a guest editor for the International Journal of Internet Technology and Secured Transactions.

Dr. Yu Chen is an Associate Professor and the Graduate Program Director at the Department of Electrical and Computer Engineering at the Binghamton University - State University of New York (SUNY). He received the Ph.D. in Electrical Engineering from the University of Southern California (USC) in 2006. His research interest lies in Trust, Security and Privacy in Smart Cities, Smart Grid, and Pervasive Computing technologies, including Cloud/Fog Computing; Smart Surveillance; Self-Powered Cyber Infrastructure based on Energy Harvesting; and Hardware Architecture for Security. He has authored or co-authored more than 100 research papers in refereed journals, conferences, and book chapters. His research has been funded by NSF, DoD, AFOSR, AFRL, New York State, and industrial partners. He has served as reviewer for NSF panels and for international journals, and on the Technical Program Committee (TPC) of prestigious conferences. He is a member of ACM, IEEE (Computer Society & Communication Society), and SPIE.

Dr. Erik Blasch is a principal scientist at the the United States Air Force Research Laboratory (AFRL) in the Information Directorate at Rome, NY, USA. From 2009-2012, he was an exchange scientist to Defence Research and Development Canada (DRDC) at Valcartier, Quebec. From 2000-2009, Dr. Blasch was the Information Fusion Evaluation Tech Lead for the AFRL Sensors Directorate - COMprehensive Performance Assessment of Sensor Exploitation (COMPASE) Center supporting design evaluations in Dayton, OH. Dr. Blasch has been an Adjunct Electrical Engineering Professor at Wright State University teaching signal processing, target tracking, and information fusion. Dr. Blasch served on the IEEE Aerospace and Electronics Systems Society (AESS) Board of Governors (BoG), was a founding member of the International Society of

Information Fusion (ISIF) (www.isif.org), and the 2007 ISIF President. He has focused on information fusion, target tracking, pattern recognition, and robitcs research compiling 650+ scientific papers and book chapters. He holds 10 patents, presented over 30 tutorials, and is an associate editor of three academic journals. His books include High-Level Information Fusion Management and Systems Design (Artech House, 2012) and Context-Enhanced Information Fusion (Spriner, 2016). Dr. Blasch received his B.S. in Mechanical Engineering from the Massachusetts Institute of Technology in 1992 and Master's Degrees in Mechanical ('94), Health Science ('95), and Industrial Engineering (Human Factors) ('95) from Georgia Tech and attended the University of Wisconsin for a MD/PhD in Neurosciences/Mech. Eng until being called to military service in 1996 to the USA Air Force. He completed an MBA ('98), MSEE ('98), MS Econ('99), and a PhD ('99) in Electrical Engineering from Wright State University and is a graduate of Air War College ('08). He is the recipeint of the IEEE Bio-Engineering Award, IEEE AESS Magazine Best Paper Award, Military Sensing Symposium Ledaership in Data Fusion Award. He is a Fellow of SPIE, Associate Fellow of AIAA, and a senior member of IEEE.

Dr. Haibin Ling received the B.S. degree in mathematics and the MS degree in computer science from Peking University, China, in 1997 and 2000, respectively, and the PhD degree from the University of Maryland, College Park, in Computer Science in 2006. From 2000 to 2001, he was an assistant researcher at Microsoft Research Asia. From 2006 to 2007, he worked as a postdoctoral scientist at the University of California Los Angeles. After that, he joined Siemens Corporate Research as a research scientist. Since fall 2008, he has been with Temple University where he is now an Associate Professor. He received the Best Student Paper Award at the ACM Symposium on User Interface Software and Technology (UIST) in 2003, and the NSF CAREER Award in 2014. He is an Associate Editor of IEEE Trans. on Pattern Analysis and Machine Intelligence and serves on the editorial board of Pattern Recognition, and served as Area Chairs for CVPR 2014 and CVPR 2016.



Dr. Genshe Chen received the B. S. and M. S. in Electrical Engineering, Ph. D. in Aerospace Engineering, in 1989, 1991 and 1994 respectively, all from Northwestern Polytechnical University, Xian, China. Currently Dr. Chen is the Chief Technological Officer of Intelligent Fusion Technology, Inc., Germantown, MD, at where he directs the research and development activities for the Government Services and Commercial Solutions. He was the CTO of DCM Research Resources LLC, Germantown, MD, and the program manager in Networks, Systems and Control at Intelligent Automation, Inc. He was a Postdoctoral Research Scientist in the Department of Electrical and Computer Engineering of The Ohio State University from 2002 to 2004. He worked at the Institute of Flight Guidance and Control of the Technical University of Braunshweig (Germany) as an Alexander von Humboldt research fellow and at the Flight Division of National Aerospace Laboratory of Japan as a STA fellow from 1997 to 2001. He did postdoctoral work at the Beijing University of Aeronautics and Astronautics and Wright State University from 1994 to 1997. His research interests include big data analytics, multi INT fusion, space situational awareness, cognitive radio for satellite communication, cooperative control and optimization for military operations, guidance, navigation, and control, Cyber Security, Internet of Things, Electronic Warfare, Game theory, Graphical theory, and Human-Cyber-Physical System.