18th International Conference on Information Fusion
Washington, DC - July 6-9, 2015

Pseudo-Real-Time Wide Area Motion Imagery (WAMI) Processing for Dynamic Feature Detection

[†]Ryan Wu, [†]Bingwei Liu, [†]Yu Chen*, [‡]Erik Blasch, [□]Haibin Ling, [§]Genshe Chen

[†]Dept. of Electrical & Computer Engineering, Binghamton University, SUNY, Binghamton, NY 13902
[‡]Air Force Research Laboratory, Rome, NY 13440
[□]Dept. of Computer & Information Sciences, Temple University, Philadelphia, PA 19122
[§]Intelligent Fusion Technology, Inc. Germantown, MD 20876

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 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 resulted from the large amount of dynamic data. Each WAMI frame is divided into multiple sub-areas and certain specifically interested sub-areas are assigned to the virtual machines in a container-based cloud computing architecture, which allows dynamic resource provisioning to meet the performance requirement. 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.

1. Introduction

Situation awareness is critical 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]. Target 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 square kilometers 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 100M of data per second, or over 400G per hour. The data scale can be even larger for high frame rate (e.g., >10Hz) and/or higher resolution videos (e.g., >10K×10K) [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 response in such systems is significantly slowed. The ability to integrate real time data to support situational awareness (SAW) target detection would be important. Inherently, effective responses for target detection rely on the level of SAW and data processing, sharing, computation, and analysis. WAMI video data has tremendous support to target detection in conjunction with other intelligence data, but it is very difficult to process and analyze the data due to its size and dynamics and security requirements [9].

Recent methods have utilized the Dynamic Data Driven Application System (DDDAS) [10], [11] for target tracking and information fusion [12], [13]. Recent examples coordinate UAVs and image sensing [14]. Liu *et al*. [15] have used the DDDAS concept to combine modeling, measurements, and software solutions for an information fusion method of tracking targets using a cloud architecture [16].

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 expected that dynamically adapts to the changing environment while minimizing the overhead at the service providers' side.

Virtualization technology for cloud computing platforms enables data security. The container-based virtualization method 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 hypervisor nor privilege instruction trapping/ translation is needed, near-native performance is achieved [42]. Another important feature is that all VMs share a single kernel. Sharing the single kernel allows for great flexibility in live resource reallocation and ultra-low overhead. We seek a container-based WAMI method.

In this paper, a pseudo-real-time WAMI data stream analysis scheme is proposed. Taking advantage of the temporal and spatial locality properties, a divide-and-conquer strategy is

adopted to overcome the challenge resulted 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 a human operator. When the operator identifies certain suspicious objects in a sub-area, the resources of the VM assigned to it are dynamically allocated to match the performance requirement. Then, the main processing engine keeps fetching new frames, dividing them, and assigning the corresponding sub-area to the VM 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 wraps up this paper with some 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) improve visual target tracking performance using background registration to compensate the camera motion. Research shows that registration can significantly improve the quality of tracking algorithms [16], [17]. However, the registration process requires tremendous computing resources, causing the entire tracking application can only achieve frame rate of less than one frame per second on a commodity computer. This is obviously not suitable for real time tracking application.

Another issue is the large data files for which researchers have addressed methods in WAMI compression [18], [19], [20]. WAMI processing requires data management [22], [23]. One example is a low-frame evaluation of WAMI tracking and performance assessment as shown by Ling *et al*. [24]. Numerous methods have been applied to WAMI, such as Sparse Representation [25], kernel learning [26], likelihood of features [27], Histogram Based Descriptors [28] to track many targets [29]. These tracking methods support activity recognition [30], context assessment [31], and enhanced situation awareness [32].

Context is an important element of WAMI analysis including spatial [33] and temporal context [34]. Context provides an assessment of the vehicle directions [35] and support pattern of life analysis [36]. The goal is to maintain Maximum Consistency Context [37]. Recent efforts have developed methods for testing [38] and real-world assessment [39]. 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 [40] 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) [41] for registration between consecutive frames before applying tracking algorithms, the quality of tracking algorithms can be significantly improved [24].

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 [42], [43] 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.

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. 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 realizable for low resolution streams, attempting to process WAMI streams in real-time consumes more computing resources than available in many systems.

It is possible to achieve real-time processing of WAMI surveillance by modifying 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 the previous method, 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 such features may pass undetected or 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 that runs it. 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. The content of the WAMI stream dictates how the hardware is utilized rather than the other hardware determining the stream processing.

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

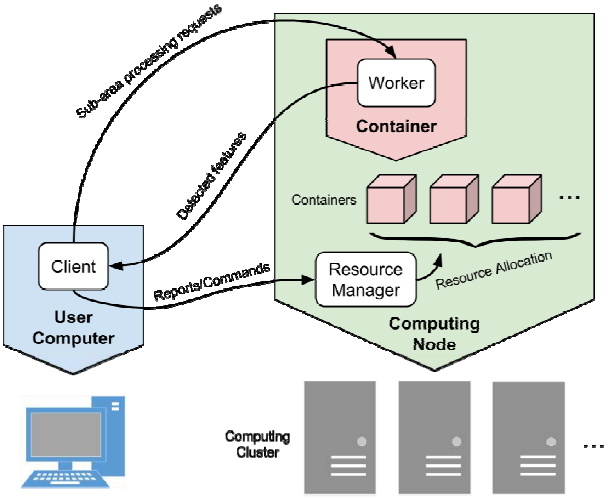

Figure 1. Pseudo-Real-Time Processing Framework.

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 sub-areas 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 TCP connections over the network. Figure 1 depicts the high level structure of the 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 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 then issues a job request to each worker that indicates a frame and sub-area along with the network information required to return detected features back to the client.

Workers wait to receive job information from clients over the network. Upon receiving a job request, the worker starts a new thread for processing the request and continues to accept additional jobs from clients. The thread 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.

Once it receives detected features from a worker, the client updates the corresponding sub-area in the user display. The client then synchronizes the sub-area to the rest of the frame, skipping frames if necessary. Since accelerated sub-areas progress through the stream faster than non-accelerated sub-areas, frame skipping is implemented to keep slower sub-areas from lagging behind. The client then sends the next request to the worker. The client also calculates the average frame rate for the worker from the time interval elapsed since sending the request. It then reports this frame rate to the resource manager responsible for the worker. In addition, 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 containers on its node. The resource manager receives both frame rate reports and acceleration commands from clients. Frame rate reports provide feedback for the proportional controller that determines the computing resources distribution between accelerated containers. Acceleration commands set the target frame rate for accelerated containers. The resource manager only updates the resource distribution client reports

(a) A WAMI Aerial Picture.

(b) SIFT Detector Key-Points on WAMI Picture.

Figure 2. WAMI Aerial Picture Sample.

and commands are received. When waiting for clients to connect, the resource manager blocks to reduce its impact on node performance.

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 Study

5.1 Experimental Setup

Our experiments are conducted on a container-based cloud computing platform at Binghamton University. The platform is comprised of three identical computing nodes. Each node has a Xeon E5-2509 quad core CPU at 2.4GHz, 16GB memory and 3TB storage. The nodes run CentOS release 6.4. The kernel is patched with OpenVZ [44] 2.6.32-042stab085.17. The containers run Ubuntu 14.04.1 LTS and are installed with OpenCV 2.4.8, which provides the scale-invariant feature transform (SIFT) feature detector and client display interface, and Boost 1.54.0, which provides multithreading support.

We tested the framework using the SIFT feature detector with WAMI data stream composed of 161 frames. 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. Fig. 2(a) is an example WAMI aerial picture and Fig. 2(b) shows the SIFT detector key-points on the picture. To isolate the effect of the framework on computing resource utilization, we assume that a copy of the data stream is stored locally in each container. Determining the optimal method for moving WAMI data streams across a network is beyond the scope of this study.

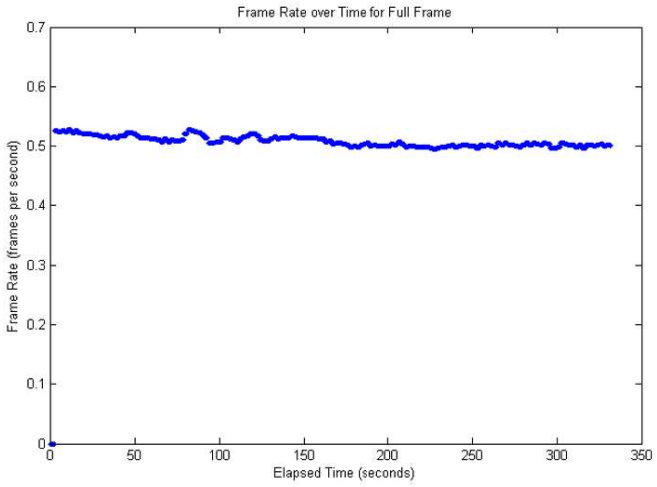

Figure 3. Frame Rate over Time for Full Frame.

5.2 Experimental Results

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

The worker's output frame rate is sampled at 100 millisecond intervals. The frame rate samples for a single pass through the data stream are plotted against time in Fig. 3. The single worker configuration requires more than 300 seconds (5 minutes) to process the data stream in full frames. The worker maintains a steady output of 0.5 frames per second (FPS) for most of the test duration.

Next, we test the effect of distributing processing work across multiple containers on a single node. As described in detail in Section 3, the framework is capable of dividing frames into sub-areas that are processed by different containers. Spatially dividing frames parallelizes the computationally intensive feature detection operation. The client divides the full frame along a grid into a number of sub-areas, each of which is assigned to a container for processing. Since all containers are located on a single node for this test, only a single resource

Table 1. Frame Rate for Varying Frame Division.

		Columns			
		1	2	3	4
Rows	1	0.49	1.01	1.48	1.95
	2	1.00	1.98	2.85	3.63
	3	1.47	2.84	3.74	3.82
	4	1.93	3.56	3.78	3.64

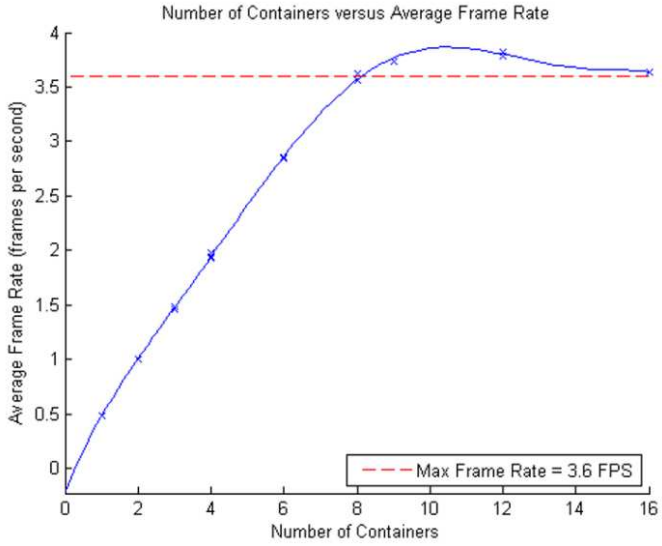

Figure 4. Number of Containers versus Frame Rate.

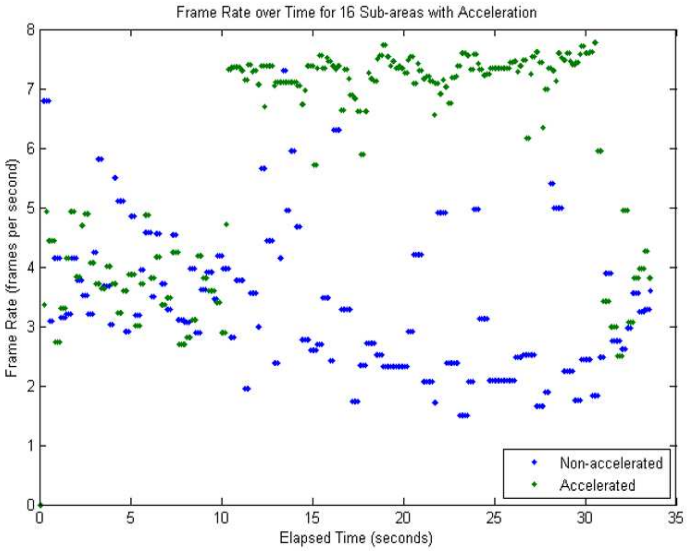

Figure 5. Frame Rate over Time with Acceleration.

manager is required to collect reports from the client. The resource manager does not redistribute resources for the duration of the test.

The framework is tested for varying degrees of frame division. In the most extreme case the full frame is divided into a 4 by 4 grid, distributing processing work across 16 containers on a single node. For each trial the WAMI stream is processed by the framework and the elapsed time is recorded. The average worker output frame rate can be calculated by dividing the elapsed time by the number of frames in the data stream. The average worker output frame rates for varying degrees of frame division is shown in Table 1. The product of the number of rows and columns is equal to the number of containers involved in the test.

The average frame rates are plotted against the number of containers in Fig. 4. The average frame rate increases linearly at a rate of approximately 0.5 FPS per container. This trend continues until 8 or more containers are used to process frames. After this turning point, the average frame rate settles to approximately 3.6 FPS.

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

The client divides the frame into 16 sub-areas, each of which is assigned to a different container. As in the previous setups, all containers run on the same node so only a single resource manager is required. This time the client issues an acceleration command for 4 sub-areas at the 10-second mark. The resource manager allocates more computing resources to the corresponding containers. At the 30-second mark, the client issues a deceleration command for each of the accelerated sub-areas. The resource manager restores the default resource allocation to the containers for the remainder of the test. The output frame rate for each worker is recorded at 100 millisecond intervals. The frame rate samples for accelerated and non-accelerated sub-areas are plotted against time in Fig. 5. Frame rate data for the other 14 sub-areas are omitted for clarity.

Before any sub-areas are accelerated, the average output frame rate is approximately 3.5 FPS. However, individual samples can vary from as low as 3 FPS to as high as 7 FPS. As the number of containers increases, variance in worker output frame rates increase. Over time, the frame rates settle closer to the average. Most samples fall within 1 FPS of the average frame rate after several seconds into the data stream. Once the acceleration commands are issued at the 10-second mark, the frame rates of the two sub-frames clearly diverge. The accelerated sub-area maintains a relatively stable frame rate above 7 FPS. By comparison, the average frame rate of the non-accelerated sub-area decreases to approximately 2.5 FPS, but occasionally higher frame rates are recorded. The frame rates for both sub-areas rapidly converge after the deceleration commands are issued at the 30-second mark. Both acceleration and deceleration manifest within 1 second of the client issuing the command to the resource manager.

6. Discussions

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

Frame division provides *significance performance gains for feature detection*. Using the same single node setup, enabling frame division with 8 or more sub-areas improves the average frame rate from 0.5 FPS to 3.5 FPS. This number of sub-areas matches the maximum number of concurrently running threads on the node. Furthermore, increasing the number of sub-areas beyond this point does not significantly reduce the average frame rate. This suggests that number of containers can be made arbitrarily large without incurring noticeable overhead,

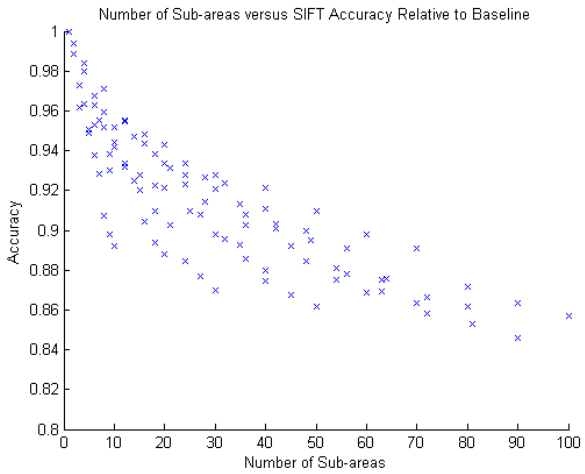

Figure 6. SIFT Accuracy Relative to Baseline.

reducing the degree of fine-tuning required to achieve optimal performance with the framework when upgrading computing cluster hardware.

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 are capable of maintaining a steady output frame rate over 7 FPS, twice the average frame rate of 3.5 FPS without redistributing resources. The results demonstrate that the high accelerated frame rates are possible because computing resources are taken away from non-accelerated sub-areas, which produce lower frame rates than normal. The framework allows for efficient utilization of limited computing resources by applying more to processing high-interest areas. 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 can be used to track targets. To assess this point, we compare the accuracy of detected features between the framework and the standalone feature detector.

We generate a list of baseline features by applying the SIFT feature detector to a frame. The same frame is divided into varying numbers of sub-areas before the SIFT feature detector is applied. The resulting features are compared with the baseline features to rate the accuracy on a scale from 0% (no matching features) and 100% (identical features). Features are considered to be matching if their coordinates are equivalent when rounded to the nearest whole pixel. Features that do not match, either missed features not detected by the framework or false features not in the reference list, are counted as errors. The accuracy is computed by taking the difference between number of baseline features and the number of errors and dividing it by the number of baseline features.

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

Figure 6 plots 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 decreases as the degree of frame division increases. The rate of decrease diminishes as the frame is divided into more sub-areas. The accuracy remains strictly above 90% when the frame is divided into less than 10 sub-areas, but numerous configurations of rows and columns can achieve similar accuracy for up to 40-50 sub-areas. This implies that the SIFT feature detector accuracy is affected by not only the degree of frame division but also the grid dimensions.

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.

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 resource is managed elastically corresponding to the computing task requirements. Such that 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) investigate more flexible and adaptive frame division methods to minimize impact on feature detection quality; 2) explore architecture level optimization that boosts the processing speed; and 3) design human-machine interface that allows human rationale be integrated into this dynamic tracking game.

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] E. Blasch, E. Bosse, D. A. Lambert, *High-Level Information Fusion Management and Systems Design*, Artech House, Norwood, MA, 2012.

[2] E. Blasch, A. Steinberg, S. Das, J. Llinas, C.-Y. Chong, O. Kessler, E. Waltz, F. White, "Revisiting the JDL model for information Exploitation," *Int'l Conf. on Info Fusion*, 2013.

[3] O. Mendoza-Schrock, J. A, Patrick, *et al.* "Video Image Registration Evaluation for a Layered Sensing Environment," *Proc. IEEE Nat. Aerospace Electronics Conf. (NAECON)*, 2009.

[4] E. Blasch, C. Yang, I. Kadar, "Summary of Tracking and Identification Methods," *Proc. SPIE*, Vol. 9119, 2014.

[5] R. Porter, C. Ruggiero, J. D. Morrison, "A Framework for activity Detection in Wide-Area Motion Imagery," *Proc. SPIE*, Vol. 7341, 2009.

[6] R. Porter, A. M. Fraser, and D. Hush, "Wide-area motion imagery: Narrowing the Semantic Gap," IEEE Signal Processing Magazine, IEEE, vol. 27, no. 5, pp. 56–65, September 2010.

[7] E. Blasch, G. Seetharaman, S. Suddarth, K. Palaniappan, G. Chen, H. Ling, A. Basharat, "Summary of Methods in Wide-Area Motion Imagery (WAMI)," *Proc. SPIE*, Vol. 9089, 2014.

[8] V. K. Asari, (ed.), *Wide Area Surveillance: Real-time Motion Detection Systems*, Section Augmented Vision and Reality, Vol. 6, Springer, 2014. http://link.springer.com/book/10.1007%2F978-3-642-37841-6.

[9] Z. H. Sun, M. Leotta, A. J. Hoogs, R. Blue, *et al*.., "Vehicle change detection from aerial imagery using detection response maps," *Proc. SPIE*, Vol. 9089, 2014.

[10] F. Darema, DDDAS Workshop Groups. *Creating a dynamic and symbiotic coupling of application/simulations with measurements/experiments*. NSF DDDAS 2000 Workshop. 2000. Available via www.1dddas.org [accessed Jan 2015].

[11] F. Darema, "Grid Computing and Beyond: The Context of Dynamic Data Driven Applications Systems," *Proceedings IEEE*, 93(3), p. 692-697, 2005.

[12] E. Blasch, G. Seetharaman, and K. Reinhardt, "Dynamic Data Driven Applications System concept for Information Fusion," *Procedia Computer Science*, Vol. 18, pp. 1999-2007, 2013.

[13] E. Blasch, G. Seetharaman, F. Darema, "Dynamic Data Driven Applications Systems (DDDAS) modeling for Automatic Target Recognition," *Proc. SPIE*, Vol. 8744, 2013.

[14] S. Ravela, "Quantifying uncertainty for coherent structures," *Procedia Computer Science*, 9:1187-1196, 2012.

[15] B. Liu, E. Blasch, Y. Chen, A. Hadiks, D. Shen, G. Chen, and A. J. Aved, "Information fusion in a cloud computing era: A systems-level perspective," *Aerospace and Electronic Systems Magazine*, IEEE, vol. 19, no. 10, pp. 16–24, 2014.

[16] M. D. Pritt, K. J. LaTourette, "Automated georegistration of motion imagery," *Applied Imagery Pattern Recognition*, 2011.

[17] Y. Wu, G. Chen, E. Blasch, H. Ling, "Feature Based Background Registration in Wide Area Motion Imagery," *Proc. SPIE*, Vol. 8402, 2012.

[18] K. Palaniappan, F. Bunyak, P. Kumar, I. Ersoy, S. Jeager, K. Ganguli, A. Haridas, J. Fraser, R. M. Rao, and G. Seetharaman, "Efficient feature extraction and likelihood fusion for vehicle tracking in low frame rate airborne video," *Intl. Conf. on Information Fusion*, 2010.

[19] A.G.A. Perera, R. Collins, A. Hoods, "Evaluation of compression schemes for wide area video," *IEEE Applied Imagery Pattern Recognition Workshop*, 2008.

[20] J. M. Irvine, S. A. Israel, "Quantifying Interpretability Loss due to Image Compression, Ch. 3 in *Video Compression*, A. Punchihewa (Ed.), InTech, 2012.

[21] P.C. Hytla, K.S., Jackovitz, E.J. Balster, J. R. Vasquez, M. L. Talbert, M.L., "Detection and tracking performance with compressed wide area motion imagery," *IEEE Nat. Aerospace and Electronics Conference*, 2012.

[22] E. Blasch, G. Seetharaman, S. Russell, "Wide-Area Video Exploitation (WAVE) Joint Data management (JDM) for Layered Sensing," *Proc. SPIE*, Vol. 8050, 2011.

[23] E. Blasch, S. Russell, G. Seetharaman, "Joint Data Management for MOVINT Data-to-Decision Making," *Int. Conf. on Info Fusion*, 2011.

[24] H. Ling, Y. Wu, E. Blasch, G. Chen, L. Bai, "Evaluation of Visual Tracking in Extremely Low Frame Rate Wide Area Motion Imagery," *Int. Conf. on Info Fusion*, 2011.

[25] Y. Wu, H. Ling, E. Blasch, G. Chen, L. Bai, "Visual Tracking based on Log-Euclidean Riemannian Sparse Representation," *Int. Symp. on Adv. in Visual Computing - Lecture Notes in Computer Science*, 2011.

[26] P. Liang, G. Teodoro, H. Ling, E. Blasch, G. Chen, L. Bai, "Multiple Kernel Learning for Vehicle Detection in Wide Area Motion Imagery," *Int. Conf. on Info Fusion*, 2012.

[27] I. Ersoy, K. Palaniappan, G. Seetharaman, R. M. Rao, "Interactive target tracking for persistent wide-area surveillance," *Proc. SPIE*, Vol. 8396, 2012.

[28] A. Mathew, V. K. Asari, "Local Histogram Based Descriptor for Tracking in Wide Area Imagery," *Wireless Networks and Computational Intelligence Comm. in Computer and Information Science*, Vol. 292, 2012, pp 119-128, 2012.

[29] J. Prokaj, X. Zhao, G. Medioni, "Tracking many vehicles in wide area aerial surveillance," *IEEE Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW)*, 2012.

[30] J. Choi, Y. Dumortier, J. Prokaj, G. Medioni, "Activity Recognition in Wide Aerial Video Surveillance Using Entity Relationship Models, 2012. In *International Conference on Advances in GIS*, SIGSPATIAL, pages 466–469, 2012.

[31] X. Shi, H. Ling, E. Blasch, W. Hu, "Context-Driven Moving Vehicle Detection in Wide Area Motion Imagery," *Int'l Conf. on Pattern Recognition (ICPR),* 2012.

[32] E. Blasch, G. Seetharaman, K. Palaniappan, H. Ling, G. Chen, "Wide-Area Motion Imagery (WAMI) Exploitation Tools for Enhanced Situation Awareness," *IEEE Applied Imagery Pattern Recognition Workshop*, 2012.

[33] P. Liang, D. Shen, E. Blasch, K. Pham, Z. Wang, G. Chen, H. Ling, "Spatial Context for Moving Vehicle Detection in Wide Area Motion Imagery with Multiple Kernel Learning." *Proc. SPIE*, Vol. 8751, 2013.

[34] P. Liang, H. Ling, E. Blasch, G. Seetharaman, D. Shen, G. Chen, "Vehicle Detection in Wide Area Aerial Surveillance using Temporal Context," *Int'l Conf. on Info Fusion*, 2013.

[35] V. Santhaseelan, V. K. Asari, "Tracking in Wide Area Motion Imagery Using Phase Vector Fields," *IEEE Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW)*, 2013.

[36] J. Gao, H. Ling, E. Blasch, K. Pham, Z.Wang, G. Chen, "Pattern of life from WAMI objects tracking based on visual context-aware tracking and infusion network models," *Proc. SPIE*, Vol. 8745, 2013.

[37] X. Shi, P. Li, W. Hu, *et al*., "Using Maximum Consistency Context for Multiple Target Association in Wide Area Traffic Scenes," *Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP),* 2013.

[38] Y. Pang, D. Shen, G. Chen, P. Liang, *et al*.., "Low frame rate video target localization and tracking testbed," *Proc. SPIE*, Vol. 8742, 2013.

[39] A. Basharat, M. Turek, Y. Xu, C. Atkins, D. Stoup, K. Fieldhouse, P. Tunison, A. Hoogs, "Real-time multi-target tracking at 210 megapixels/second in wide area motion imagery," *IEEE Winter Conf. on Apps. of Computer Vision* (WACV), 2014.

[40] D. G. Lowe, "Distinctive image features from scale-invariant keypoints." *International journal of computer vision* 60.2 (2004): 91-110.

[41] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." Communications of the ACM 24.6 (1981): 381-395.

[42] R. Wu, Y. Chen, E. Blasch, B. Liu, G. Chen, and D. Shen, "A container-based elastic cloud architecture for real-time full-motion video (FMV) target tracking," *Applied Imagery Pattern Recognition Workshop (AIPR), 2014 IEEE*, vol., no., pp.1,8, 14-16 Oct. 2014

[43] 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.

[44] SWSoft, "Openvz server virtualization," http://www.openvz.org/, 2006.