SSC21-V-07

Developing a Prototype Ground Station for the Processing, Exploitation, and Dissemination of pLEO Sensor Data

Jennifer Wilbur, David Simenc, Eric Principato, Travis Williams,
Jason Hamant, Matthew McHugh, Sander Malmquist, John Maloney
SciTec, Inc.
100 Wall Street, Princeton, NJ 08540; 609-921-3892 x355
jdavis@scitec.com

Debi Rose, Dan Rossiter, Paul Wood
Southwest Research Institute
1050 Walnut St #300 Boulder, CO 80302; 303-324-4704
debi.rose@boulder.swri.edu

ABSTRACT

The Air Force's Space and Missile Systems Center (SMC) recently executed a quick-turnaround (16 month) effort through the Defense Innovation Unit to develop a prototype ground architecture demonstrating low-latency processing, exploitation, and dissemination of data collected by notional multi-phenomenology sensors hosted on small satellites in a proliferated LEO constellation. This effort, led by the Southwest Research Institute and supported by teammates, Amazon Web Services, SpaceX, and SciTec, Inc., involved the modeling and simulation of a variety of different OPIR, EO/IR, and SAR data streams; transporting these data via space and ground networks; processing the data in the AWS cloud environment; and then disseminating resulting products to tactical users. In this paper, we present an overview of the data transport and mission data processing, performance results from the application of our various Mission Data Processing Chains, a summary of our findings on the latencies associated with both data transport and data processing, and lessons learned including insight into ground-based vs. on-board processing.

1. INTRODUCTION

The Southwest Research Institute (SwRI) and teammates SciTec, Amazon Web Services (AWS), and SpaceX recently delivered a novel, commercial processing, exploitation, and dissemination prototype to SMC. The objective was to demonstrate a low-latency, horizontally-scalable, PED capability featuring cloud-based processing for data collected by future payloads sensing in multiple modalities hosted on commercial spacecraft and downlinked through commercial gateway injection points.

The intent for this prototype was to deliver processed data in formats usable by tactical users deployed to forward operating locations. The delivered prototype addressed a lack of established gateways or processes to ingest data collected from DARPA's BLACKJACK-capable spacecraft and distribute that data through a commercial gateway and seamlessly deliver it to a location in theater that needs it most to meet critical timelines without significant human-machine interface and latency. The SwRI PED prototype effort resulted in the generation of a process recommendation, along with an associated hardware and software solution using Overhead Persistent Infrared (OPIR), Synthetic Aperture Radar (SAR) and EO/IR image data as test cases delivering capability to users in any Continental United States (CONUS) or forward operating location.

In the following sections, we detail each of the three (OPIR, SAR, and EO/IR) demonstrations.

2. OPIR PROTOTYPE DEMONSTRATION

For the OPIR demonstration, two scenarios were considered: 1) Collection by three OPIR sensors collecting frame data with 2,048 x 2,048 pixels at 20 Hz on an area of interest with a simulated raid of missile threats and 2) Collection by a single OPIR sensor with 4.096 x 4,096 pixel frames at 10 Hz. The first scenario was intended to exercise the mission data processing pipeline, which includes sensor specific processing – i.e., background suppression, detection, and 2D tracking – as well as track correlation and fusion to generate 3D tracks. The second scenario was intended to expose any potential latencies in data transport and sensor-specific processing.

The top level configuration of the primary OPIR 2k x 2k sensor prototype scenario is shown in Figure 2-1. Simulated sensor data is generated for the three sensors

– a single aft mounted sensor on one simulated pLEO spacecraft and a fore and aft mounted sensor set on a second simulated pLEO spacecraft.

In this scenario, we used the Amazon Web Services (AWS) Ground Station model for a ground antenna site capable of receiving a downlink from a Low Earth Orbit (LEO) satellite. This concept co-locates an AWS Ground Station antenna with an AWS Data Center. This model allows the data received from a satellite to use data center resources for processing and transport initiation.

The 2k x 2k OPIR prototype used the following configuration:

- Data for casino1-aft sensor initiated in us-east-2 (Ohio) AWS region
- Data for casino2-aft, casino2-fore sensors initiated in us-west-2 (Oregon) AWS region (non-GovCloud)
- Data from all sensors flowed independently into a Simple Storage Service (S3) bucket in the us-west-2 (Oregon) region and transferred to the us-gov-west-1 (Oregon) AWS GovCloud region
- Final processing of satellite data back into raw format, transfer of data to MDP, and MDP processing in us-gov-west-1 (Oregon) AWS GovCloud region.

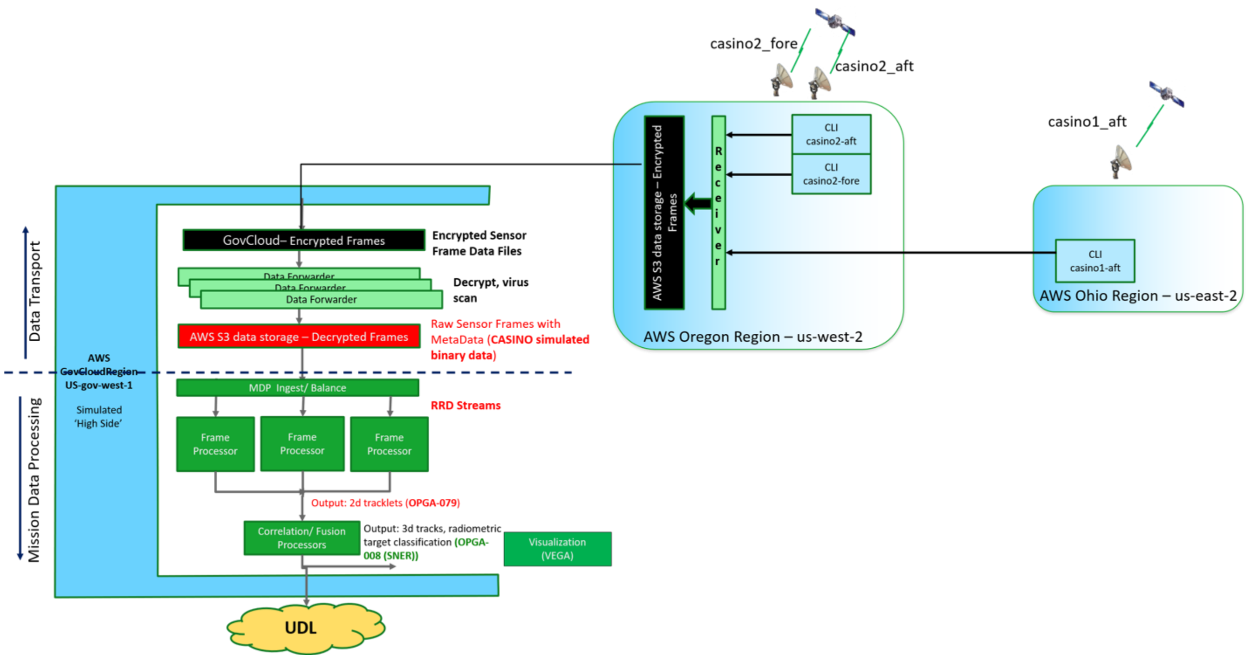

casino2_fore
casino2_aft
casino1_aft
AWS S3 data storage – Encrypted Frames
Receiver
CLI casino2-aft
CLI casino2-fore
AWS Oregon Region – us-west-2
CLI casino1-aft
AWS Ohio Region – us-east-2
Data Transport
Mission Data Processing
AWS GovCloudRegion US-gov-west-1
Simulated 'High Side'
GovCloud– Encrypted Frames
Encrypted Sensor Frame Data Files
Data Forwarder
Data Forwarder
Data Forwarder
Decrypt, virus scan
AWS S3 data storage – Decrypted Frames
Raw Sensor Frames with MetaData (CASINO simulated binary data)
MDP Ingest/ Balance
RRD Streams
Frame Processor
Frame Processor
Frame Processor
Output: 2d tracklets (OPGA-079)
Correlation/ Fusion Processors
Output: 3d tracks, radiometric target classification (OPGA-008 (SNER))
Visualization (VEGA)
UDL

Figure 2-1. Three OPIR Sensor Demonstration Overview

This initial prototype scenario was derived to provide a 'stress test' on the ability of the commercial ground resources to transport and then process all data frames from the on-orbit OPIR sensors. In this stress scenario – data sets representing all data collected by the OPIR sensors are moved through the commercial network, prepared for processing, and then processed in a secure cloud MDP region. Adding to the 'stress' of the 3-sensor scenario – data from multiple sensors will be received at different ground stations and must be transferred to a common secure cloud processing site within a small time window (less than approximately 5 seconds) to ensure all data associated with each time slot can be properly processed and correlated.

2k x 2k OPIR Demonstration – Data Transport

The raw sensor data frames – including 2k x 2k 16 bit data frames along with associated meta-data – were staged in files. Each file contained a one second sensor data collection, which resulted in 20 frames of the sensor plus meta data frames in a single file. This approach creates a 160 Mbyte file to represent each second of data. For this scenario, 165 seconds of data for each sensor was transported and processed. The 165 second time was selected to simulate filling a potential LEO downlink period for a spacecraft in a 1,000 km orbit.

In order to simulate the need to protect the data during the downlink and then transport through non-classified regions, the data files for the prototype were encrypted. Files were also encoded in Consultative Committee for

Space Data Systems (CCSDS) protocol to simulate a common protocol used for spacecraft to ground communications. For the prototype, a commercial encryption method was used.

Encrypting and encoding the raw data files used in the prototype demonstration occurred prior to initiating the transfer of the files through the commercial network to a secure cloud processing location. This order of events was selected since the encryption and encoding activities would typically occur on the spacecraft and would not contribute to latencies associated with ground transport and processing activities.

Prototype execution and timing measurements were initiated as each file, containing a 1 sec data set, started the data transport process - which includes decommutating the CCSDS encoding, transporting the file from the origin location to a designated secure cloud processing location, and, once there, virus scanning, decrypting and passing the data on to an MDP processing suite of applications.

Data files for each individual sensor were staged to support the 3-sensor prototype configuration. Data flow for each of the three file sets was configured to be initiated via separate tasks staged in Docker containers in selected AWS regions.

2kx2k OPIR Demonstration – Mission Data Processing

The Mission Data Processing (MDP) architecture utilizes Amazon Web Services (AWS) to host mission processing applications for Overhead Persistent InfraRed (OPIR), Synthetic Aperture Radar (SAR), and Electro-Optic/InfraRed (EO/IR) mission data threads. The MDP applications form a micro-service oriented architecture of advanced mission processing algorithms deployed as containerized, elastic services in AWS to provide low-latency, high accuracy data exploitation. The applications are deployed through Infrastructure as Code (IaC) onto AWS Elastic Compute Cloud (EC2) instances including multiple Compute Optimized (c5) and Accelerating Computing (g4) AWS EC2 virtual machines. The MDP IaC – consisting of Cloud Formation scripts and templates - encompasses all activities required to provision, deploy, and execute the MDP environment. Autonomous MDP orchestration dynamically scales resources and applications based on processing load. All MDP mission threads consist of object storage in AWS Simple Storage Service (S3), mission specific EC2 instances for processing, and messaging services providing data over a defined network port for visualization applications to view data in near-real-time.

The 3 sensor OPIR MDP environment consists of multiple Compute Optimized (c5) AWS EC2 virtual machine instances. Applications are distributed to balance network and computational requirements. By using a micro-service-based architecture, multiple applications are easily distributed among multiple instances as opposed to requiring very large individual EC2 instances sized for the highest anticipated processing and network loads. Our architecture further includes the MDP Application Analysis Dashboard which provides visualization, system monitoring, and data interrogation of the MDP applications in real-time. A functional flow diagram of the 2k x 2k OPIR processing chain is depicted in Figure 2-2.

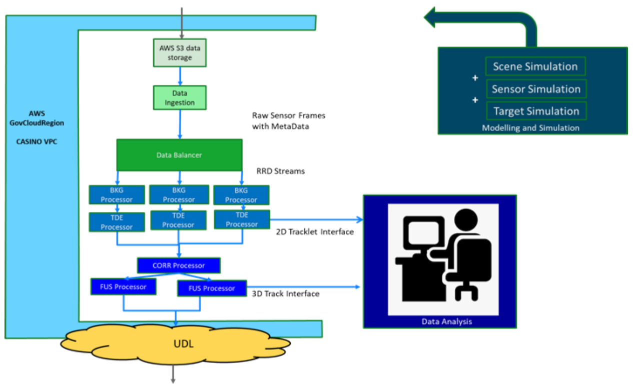


Figure 2-2. Functional flow for the 2k x 2k OPIR mission data processing chain

2k x 2k OPIR Data Simulation

The OPIR mission data simulation was developed to allow for testing of the data transport architecture, while also containing realistic sensor and target motion as well as sensor noise in order to test the MDP components of the ground architecture. The 3 sensor dataset contains 165 seconds of 20 Hz simulated OPIR frames and metadata for two pLEO satellites; one satellite has its fore and aft sensors simulated while the other has just the aft. The sensors were simulated as body-fixed 2,048 by 2,048 pixel starers operating as part of the same orbital plane and were spaced out to provide maximum stereo coverage for downstream processing. Scenes contained 14 ballistic targets of different range classes, and two dim, constant altitude (non-ballistic) targets.

Scenario geometry and CONOPs were simulated in SciTec's internally developed PACMAN tool, which contains modules for simulating orbits, sensors, and targets. For the OPIR scenarios, sensors with CASINO orbits and body-fixed pointing were placed to maximize sensor overlap, and targets were inserted into the 3D scenario. The resulting per-frame geometry was then run through PRA Toolkit v2. PRA Toolkit, developed by Photon Research Associates, uses atmospheric radiative transport models such as MODTRAN (v4.0) and

MOSART (v1.60) as well as databases of cloud and terrain material fractions, altitude maps, and spectral responses to generate realistic composite scene images. Once the scene images were generated in PRA Toolkit, the original PACMAN geometry was used to add the targets into the scenes. Finally, Python scripts were used to add sensor noise, digitize the frames, and convert them, along with their associated metadata to the final binary format.

Mission Data Processing Chain

The mission data processing chain consists of background suppression, mono-track generation, correlation, and fusion services. Raw frame data is fed into the system and farmed to background suppression services on a per-sensor basis. Background suppressed frames are then passed to the mono-track generation service, which performs target detection, exceedance chip extraction, and track before detect tracking on the suppressed frames. This results in 2D tracks that are then correlated. The resulting correlated tracks are passed to the fusion processor to generate full state estimates. The final state estimates are messaged to tactical users and can be compared to the truth input tracks through an App Analysis Dashboard, shown in Figure 2-3

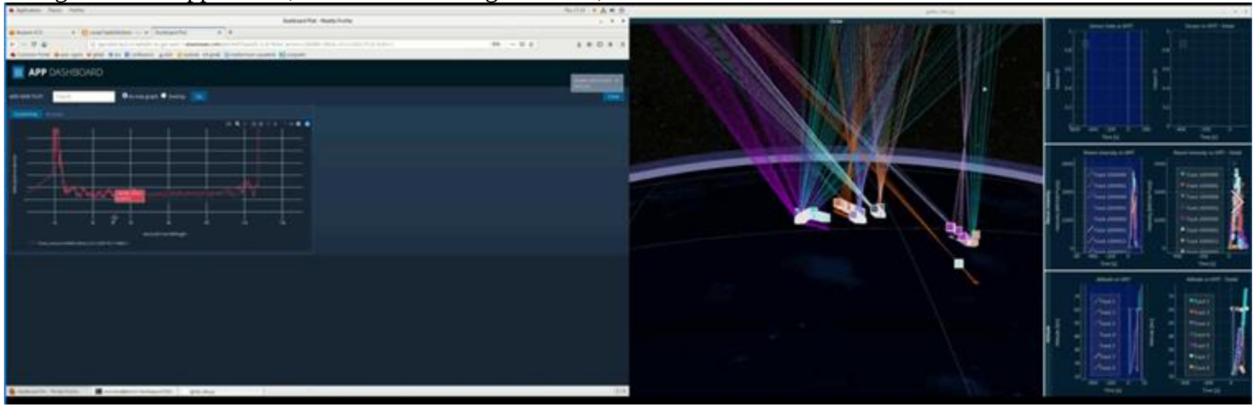


Figure 2-3. Screen captures of the App Analysis Dashboard and 3D global track visualizer

The background suppression service and mono-track generation services are derived from SciTec's ARROW application and are tuned for the CASINO dataset simulations. The service consists of many tunable parameters, which allow an operator to select and tune the algorithms that are used for suppression, detection, extraction, and tracking. These parameters were selected with the constraint that they must be able to run at 20 Hz for the 2k x 2k frames in the cloud processing environment. Otherwise, parameters were tuned to detect the targets as early in their trajectories as possible and to continue following tracks for as long as possible, ideally well into the second stage burns.

4k x 4k OPIR Demonstration

The data transport scenario for this demonstration was similar to that of the 2k x 2k OPIR demonstration.

The 4k x 4k OPIR prototype used the following configuration:

- Data for casino 4kx4k sensor initiated in us-east-2 (Ohio) AWS region
- Data flowed into an S3 bucket in the us-west-2 (Oregon) region and transferred to the us-gov-west-1 (Oregon) AWS GovCloud region
- Final processing of satellite data back into raw format, transfer of data to MDP, and MDP processing in us-gov-west-1 (Oregon) AWS GovCloud region

The raw sensor data frames – 4k x 4k pixel, 16 bit data frames along with associated meta-data – were staged in files. Each file contained a two second sensor data collection, which results in 20 frames (10 frames each second) of the sensor plus meta data frames in a single file. This approach creates a 640 Mbyte file to represent each two seconds of data.

As in the case of the 2k x 2k data, the data files for this prototype were encrypted and encoded using the CCSDS protocol. Data flow for the file set was configured to be initiated via a task staged in a Docker container in a selected AWS region.

4k x 4k OPIR Data Simulation

Frames and metadata were simulated for the 4k x 4k demo in a manner similar to the 3 sensor 2k x 2k demo. SciTec's PACMAN tool was used to create the 3D scene geometry including sensor and targets; that geometry was used to populate the PRA Toolkit input files (one is needed for each time point); and the PRA Toolkit output scene radiance frames were combined with target signatures and sensor noise to create frames in the final binary output format. Because the PRA Toolkit simulations are a time intensive process that scales with sensor size, only 200 frames were simulated for the 4k x 4k sensor. They were simulated at 10 Hz instead of 20 Hz in order let the simulation cover a larger simulation time period.

This simulation has some differences from the 3 sensor OPIR simulation. In terms of sensor geometry, the single 4k x 4k sensor in this dataset is staring along the body-fixed nadir direction, with a limb-to-limb 124 degree field of view. The other difference is that while the same targets as the 3 sensor simulation were used in this simulation, the targets had their timing shifted to start earlier so that as many targets as possible were active and boosting during the smaller 20 second window that was simulated for the 4k x 4k scenario. An example 4k x 4k image with target trajectories overlaid is shown in Figure 2-4

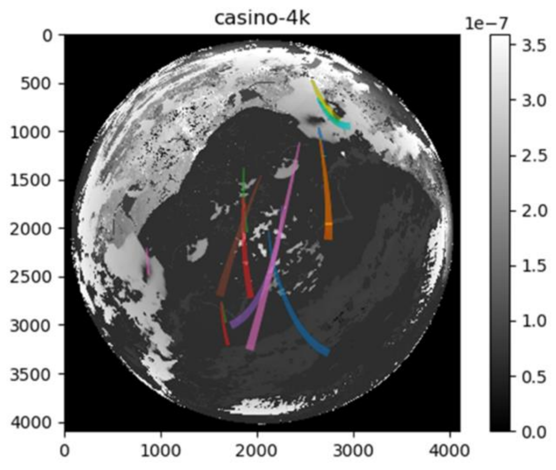


Figure 2-4. Example 4k x 4k OPIR background frame with target trajectories overlaid

The mission data processing pipeline used the same steps as in the 3 Sensor, 2k x 2k pixel OPIR Demonstration. Because data were only simulated for a single sensor instead of 3 sensors (thereby providing stereo coverage), this demonstration was meant primarily to test data flow and the background suppression and mono-sensor tracking components of the processing architecture.

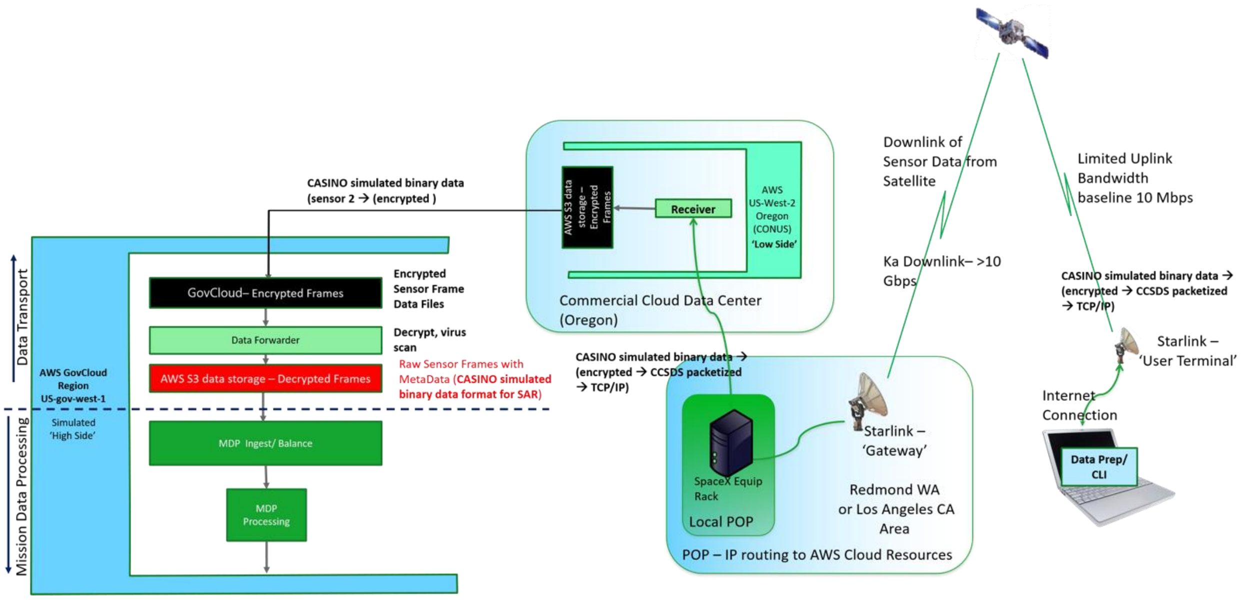


Figure 2-5. Overview of SAR Demonstration Scenario

3. SYNTHETIC APERTURE RADAR (SAR) DEMONSTRATION

The top level configuration of the SAR sensor prototype scenario is shown in Figure 2-5. In contrast to the OPIR prototypes, the SAR prototype was demonstrated by initiating the simulated data from a Docker container residing on a personal computer connected to the internet via a Starlink connection.

The SAR prototype used the following configuration:

- Data for SAR sensor initiated in personal computer Docker container connected to a Starlink internet connection
- Data flowed into an S3 bucket in the us-west-2 (Oregon) region and transferred to the us-gov-west-1 (Oregon) AWS GovCloud region
- Final processing of satellite data back into raw format, transfer of data to MDP, and MDP processing in us-gov-west-1 (Oregon) AWS GovCloud region

In addition to testing the SAR prototype using the Starlink configuration, tests initiating the SAR data in the AWS us-east-2 (Ohio) region were executed to assess the transfer speed that could be obtained through the use of a high speed connection.

SAR Data Simulation

The standard approach for simulating synthetic aperture radar (SAR) signals over complex scenes relies on constructing a set of point reflectors. Point reflectors model the radar response over a scene and received In-phase / quadrature (I/Q) signals are computed along the aperture. For complex or extended scenes where a radar response is required for every resolution cell in the scene, this technique can be prohibitively expensive. Axelson proposes a method for simulating I/Q for stripmap collected SAR signals circumventing a great deal of the complexity involved in the standard approach[1]. In this approach, a fully processed SAR image is directly simulated, and passed through an inverted processor to recover an I/Q signal. We extend Axelson's method to the case of Spotlight collection mode.

The scene begins with a digital surface map (DSM) and greyscale image of a two square nautical mile region in Trento, Italy. The DSM has a true resolution of 1 m x 1 m, and the greyscale image 20 cm x 20 cm. Both are resampled to 30 cm x 30 cm resolution. The DSM facilitates a three dimensional geometry model. Elevation values are leveraged to determine which resolution cells are shadowed, and which ones exhibit foreshortening and layover effects. In Figure 3-1, the direction of the radar pulse is indicated by the arrows in the top left, where the incidence angle is defined as the angle between the radar's line of sight and the sensor's nadir. The count of resolution cells contributing to each pixel in slant range are indicated at the top of the figure.

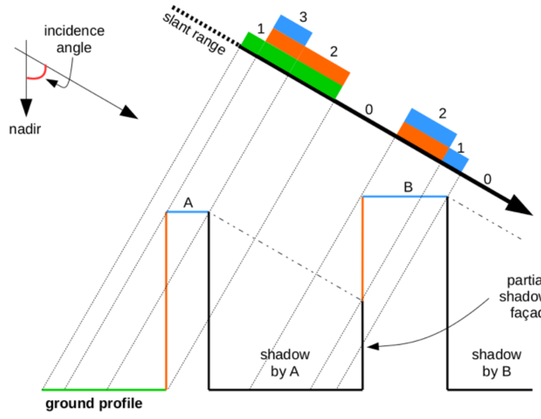


Figure 3-1. SAR resolution example

A greyscale image is combined with a set of prior distributions over terrain types to estimate the actual radar returns for each resolution cell in the image. Skolnik proposes a set of distributions for estimating radar returns based on incidence angle, terrain type (lake, city, forest, or farmland), and carrier frequency using real data produced by Sandia Corporation[2]. We construct a similar distribution visible light reflection over our Trento scene by sampling pixels from the greyscale image and assigning terrain types by hand. This allows us to estimate a mapping between the visible light distribution and the radar distribution. That mapping is applied to every pixel in the greyscale image, yielding an estimated radar return for each resolution cell in our scene. These values are combined with the geometric model above to produce an image–this image resembles a fully processed SAR signal with perfect noise mitigation.

The images in Figure 3-2 illustrate the SAR image formation process. The image on top is a DSM of a small region in Trento. The image on the bottom shows the resulting SAR image, where shadows and layovers are clearly visible, evidence of the slant range geometry model. To reconstruct the SAR signal, the simulated SAR image is passed through an inverted processor. The inverted processor is the mathematical inverse of the range migration algorithm (RMA) detailed in the following section. Phase noise and jitter are incorporated into the inverse processor to improve the authenticity of the recovered raw I/Q.

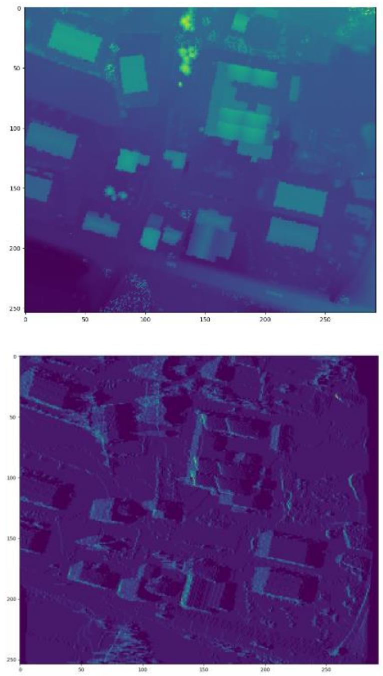


Figure 3-2. SAR image formation example

SAR Mission Data Processing

RMA is well suited to imaging large scenes at high resolution. Carrara points out that "RMA images do not suffer from the space-variant defocusing and geometric distortion that wavefront curvature induces with the use of the [polar format algorithm]"[3]. One drawback of RMA is that it requires a high along-track sampling rate. The input for RMA is a SAR signal after motion compensation and range deskew, which are considered to be preprocessing operations. A block diagram of the processing sequence is shown in Figure 3-3.

The along track Fourier transform translates the SAR signal into the spatial frequency domain, which facilitates matched filtering. The matched filter applies a two dimensional phase compensation, which perfectly corrects the range curvature of all scatterers with range equal to that of the scene center. Stolt interpolation then compensates the range curvature of all scatterers by an appropriate warping of the SAR signal data. Finally, a two dimensional inverse Fourier transform compresses the signal in both range and azimuth to recover a fully processed image.

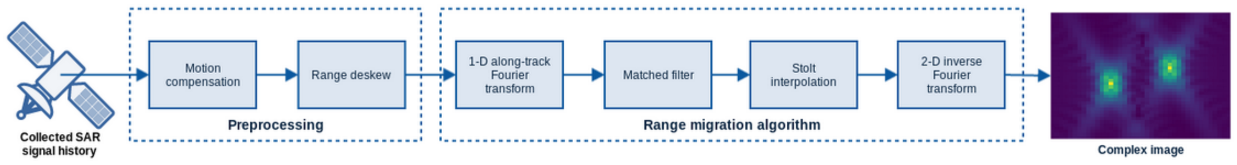

Collected SAR signal history
Motion compensation
Range deskew
Preprocessing
1-D along-track Fourier transform
Matched filter
Stolt interpolation
2-D inverse Fourier transform
Range migration algorithm
Complex image

Figure 3-3. Block diagram of SAR processing

The Mission Data Processing pipeline was designed to handle uptake of data from remote sensors, and subsequent processing and visualization of the data. We took advantage of the Infrastructure-as-Code architecture, including containerized applications, to create a dynamically deployable application suite that is appropriate for any operating system with access to gnome terminal and python3.

The Mission Data Processing data transfer application toolkit is designed to handle data transfer between AWS storage and compute services, processing the Mission Data on that compute service, and finally transferring the data to a local workstation for visualization. These three action areas are activated by a single IaC script by the local user, and the remote processing applications are containerized via docker images. The processing application could, in principle, be extended to kickoff processing when images are detected by a secondary application. The docker images are stored on the elastic container registry (ECR) on AWS and protected by the security credentials of the user group.

Upon the execution of the IaC script, an AWS EC2 instance is initialized. This instance is pre-loaded with the libraries necessary to load the Docker images that contain the app and is tied to an elastic block storage (EBS) resource for data storage infrastructure. The EBS resource used for our demonstrations had 200 GB, but is generally configurable to the amount desired. The SAR processing application and the required Python libraries constituted a docker container that was less than 1.5 GB in size, leaving a large block of storage available for the data to be processed.

When the EC2 instance is initialized, the Docker image is pulled from ECR and containerized, and the remote Mission Data Processing application is executed. This application looks for SAR data in a specified address in the AWS storage service S3 and moves the data from S3 to the EC2 Docker container, where it is processed and the results are published to a user-specified network socket. Data streaming is possible, but would be limited by transport time and processing time.

The IaC local service determines if the remote compute resources are initialized by attempting to SSH into them (the EC2 instance). Once the IP is accessible, the other local application terminals are initialized. There are two applications that run locally, in conjunction with the remote containerized application: These are a logging output window and the Mission Data Receiver-Visualizer. The logging .output is fetched from the containerized app on AWS, and is the user's gateway to seeing the output from the remote app. Information is published to this logging terminal that reports when any of the following things happen: 1) new SAR data is found in a target AWS S3 storage bucket, 2) data is moved from S3 to EC2, 3) data is being processed by the SAR decoding application, and 4) data is being sent down to the local machine.

The second window is the data receiver and visualization application. This application listens to the network socket that the remote application publishes to and creates a visualization of the processed SAR image for the local user. The visualizer displays the raw test image, an image of the landscape, and the processed SAR image.

4. EO-IR AUTOMATED TARGET RECOGNITION DEMONSTRATION

Similar to the OPIR prototypes, the EO/IR prototype was demonstrated by initiating the simulated data from a Docker container residing in a selected AWS region.

The EO/IR prototype used the following configuration:

- Data for EO/IR sensor initiated in the me-south-1 (Bahrain) region
- Data flowed into an S3 bucket in the us-west-2 (Oregon) region and transferred to the us-gov-west-1 (Oregon) AWS GovCloud region
- Final processing of satellite data back into raw format, transfer of data to MDP, and MDP processing in us-gov-west-1 (Oregon) AWS GovCloud region

In addition to testing the EO/IR prototype using the Starlink configuration, tests initiating the EO/IR data in the AWS us-east-2 (Ohio) region were executed to assess the transfer speed that could be obtained through the use of Continental United States (CONUS) connection.

EO-IR Data and Mission Data Processing

An algorithm for automatic target recognition (ATR) in EOIR images was developed. This algorithm uses a combination of RetinaNet[4] and U-NET[5] machine learning architectures to automatically identify targets in input images. The RetinaNet algorithm was trained on the xView[6] dataset and tested on both xView and SkySat[7] datasets. The U-NET algorithm was trained initially with Landsat data, and then applied to the xView dataset using transfer learning.

RetinaNet is the machine learning algorithm primarily responsible for ATR. The detections by RetinaNet are comprised of a bounding box (x,y coordinates describing a rectangle around the object), a class label, and a confidence score denoting how confident the algorithm is in correctly identifying the target. RetinaNet was found to have exceptional performance in identifying targets, but there were significant false positive detections in clouds. Therefore, U-NET was added to supplement RetinaNet. U-NET is a classification algorithm that generates a class label for every individual pixel in an image and was used to detect clouds and create a penalty function for detections in clouds. The combination of RetinaNet and U-NET led to accurate detections while minimizing false positives.

A depiction of the mission data processing architecture developed for the EO/IR demonstration is provided in Figure 4-1.

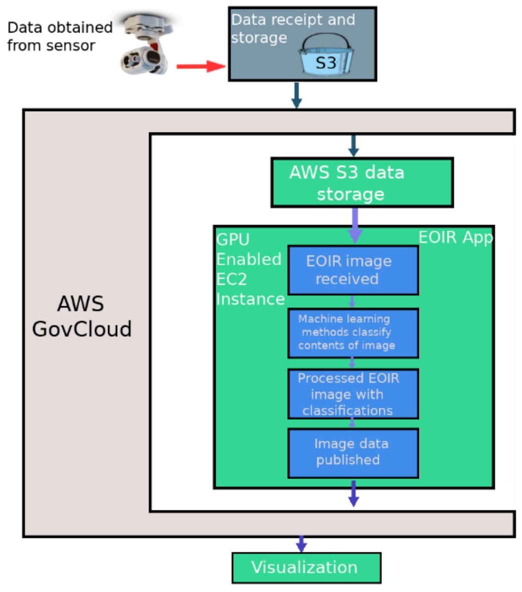


Figure 4-1. EO/IR Mission Data Processing architecture

The Mission Data Processing data transfer application toolkit for EOIR is the same as for what that outlined for SAR processing above. On the kickoff of the IaC script, an AWS EC2 instance is initialized. This instance is pre-loaded with the libraries necessary to load the Docker images that contain the app and is tied to an EBS resource for data storage infrastructure. The EBS resource used for our demonstrations had 200 GB, but is generally configurable. The EOIR processing application and the required Python libraries constitute a docker container of approximately 15 GB in size. The EOIR container is so much larger than that used for the SAR demonstration due to the machine-learning backbone of the application that requires large libraries for GPU processing.

When the EC2 instance is initialized, the Docker image is pulled from ECR and containerized and the application is executed. This application looks for EOIR data in a specified address in the AWS storage service S3 and moves the data from S3 to the EC2 Docker container, where it is processed and results are published to a user specified network socket.

Once the IP of the EC2 instance is accessible, the other local application terminals are initialized. Two applications run locally in conjunction with the remote containerized application. These are a logging output window and the Mission Data Receiver-Visualizer. The logging output is the user's gateway to seeing the output from the remote app and is fetched from the containerized app on AWS. Information is published to this logging terminal, such as reports when: 1) new EOIR data is found in a target AWS S3 storage bucket, 2) data is moved from S3 to EC2, 3) data is being processed by the EOIR decoding application, and 4) data is being sent down to the local machine.

The data receiver and visualization application listen to the network socket that the remote application publishes to and creates a visualization of the processed EOIR image for the local user. That image contains four subpanels – one of the original image, one of the original image plus object identification, one of a cloud mask, and the fourth containing a combination of the cloud mask and target objects with an associated identification confidence score.

A screen capture from the EO/IR demonstration is shown in Figure 4-2

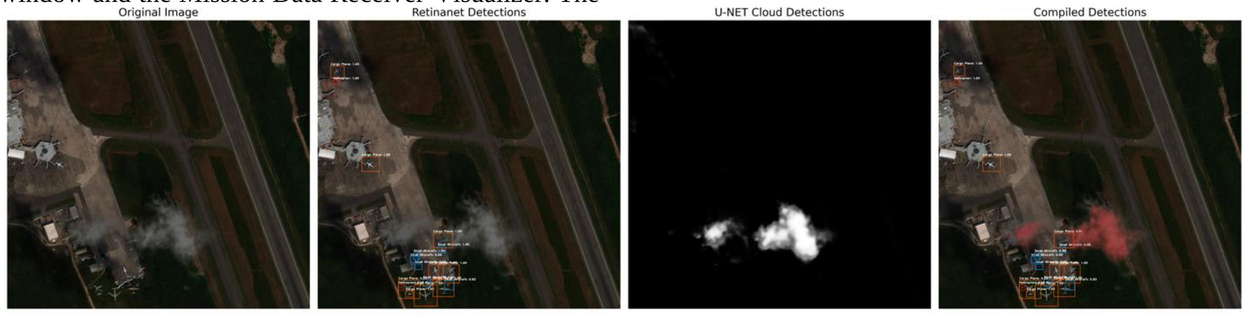


Figure 4-2. Visualization of the output of the EO/IR ATR application

Training the RetinaNet ATR algorithm

The RetinaNet model was trained using the xView dataset to detect aircraft and boats. The original xView dataset includes 846 labelled images; however, only a subset of this dataset was used to train the model. The subset was determined by removing any images that did not contain at least one aircraft or boat annotation, leaving a total of 300 labelled images. These images were further divided into a training set (237 images) and a validation set (63 images) via random selection. The training set was used to train the weights of the RetinaNet model, while the validation set allowed for performance evaluation after each training step. Models were trained between 200 and 500 steps, and the model with the highest validation performance was selected for use in the prediction model. In addition to the 300 labeled images, 281 unlabeled images were utilized as a test set to further validate the model performance. Since the dataset was unlabeled, accuracy of the model using the validation set was determined through visual inspection. Finally, 13 unlabeled images from the SkySat dataset were used to further probe the accuracy of the model and determine its performance across datasets.

A subset of labelled targets provided with the xView dataset were selected in an effort to reduce the number of classes included in the initial training sequence, thereby reducing the amount of training required. Initially, only aircraft targets were selected, comprising 4 total target classes (Cargo Plane, Small Aircraft, Fixed-Wing Aircraft, and Helicopter). Later, boat targets were added to ensure that RetinaNet was capable of learning new targets. The addition of boats increased the total target class count to 14.

Due to computational constraints, an image pre-processing algorithm was designed to segment images into smaller subimages prior to training and evaluation in RetinaNet. The xView dataset images were typically large, with sizes around 3,000 x 3,000 pixels. An image of this size, after convolution operations within RetinaNet, exceeds the memory capacity of most available GPUs. By segmenting the image into multiple subimages, the computational requirements for processing images through RetinaNet are reduced. The subimage size is a tunable parameter, allowing the user to control the tradeoff between computation requirement and training time.

Model training was performed in such a way that every subimage was processed by the model during each training epoch. A subimage size of 512x512 pixels was used, resulting in 16,680 total subimages for the training dataset. Using a 10 GB GPU, subimages were able to be processed in batches of 4, leading to 4,170 steps per training epoch. Each step required an average of 400 ms to process, leading to a total time requirement of 28 minutes per epoch. Models were typically trained for between 200 and 500 epochs; therefore, training time for a model required a total of 4-10 days.

A non-maximum suppression (NMS) algorithm was added at the end of the RetinaNet algorithm to handle multiple detections of the same object. Multiple detections became very prolific due to the subimage strategy described above, since some targets appeared in multiple images and were detected multiple times. NMS is a technique which keeps the detection with the highest confidence score while removing all other detections. An example of the effects of implementing the NMS may be seen in Figure 4-3, where the number of detections is significantly reduced.

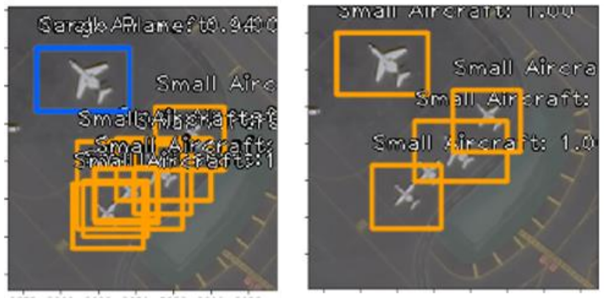


Figure 4-3. Comparison of raw RetinaNet detection and NMS

One of the largest drawbacks discovered with RetinaNet was significant false positive detections in images containing clouds. Examples of these false positive detections may be seen in Figure 4-4. In order to combat these false positives in RetinaNet, a second algorithm was implemented into the EO/IR framework: U-NET. U-NET is a classification algorithm that identifies a class category for every pixel in an image, which makes it well suited for cloud detection within EO/IR data.

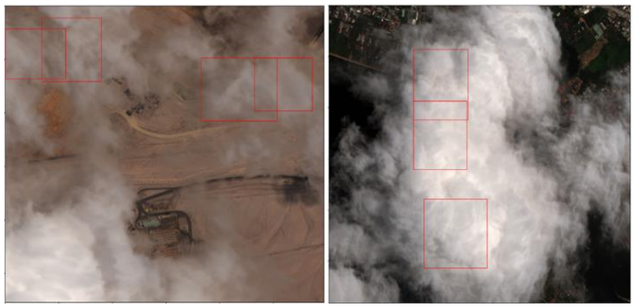


Figure 4-4. False positive detections (shown in red) within clouds with no targets present

In order to accurately train U-NET for cloud detection on the xView dataset, transfer learning was employed from a different, cloud-labelled dataset. A series of Landsat8 datafiles[8] archived at SciTec was used to initially train U-NET for cloud detection. Upon completion of training this model on Landsat8, the trained model was then used to detect clouds in the xView dataset. Initial detections on the xView dataset were poor, however. Clouds were detected well, but many non-clouds, such as buildings and roads, were also being detected as clouds. From visual observation, it appeared that U-NET was detecting the brightest object in xView images and classifying it as clouds.

To mitigate the errors in transferring U-NET between Landsat8 and xView, a ground truth for the xView dataset was developed. Initially, 8 images containing clouds were selected and a cloud mask was drawn for the image, by hand, in Microsoft Paint. These manually constructed cloud masks were provided to U-NET with the weights from the Landsat8 dataset, with the hope that additional data on xView would help level out the erroneous cloud detections. Using the newly trained U-NET model, 10 additional xView images were predicted. These had significantly better cloud detections, but still had some errors. The errors were corrected by hand, provided back to the U-NET model as training data, and the process was repeated. After generating and correcting 100 images from xView, the model results were deemed sufficiently accurate for detecting clouds. An example from the fully trained model is shown in Figure 4-5

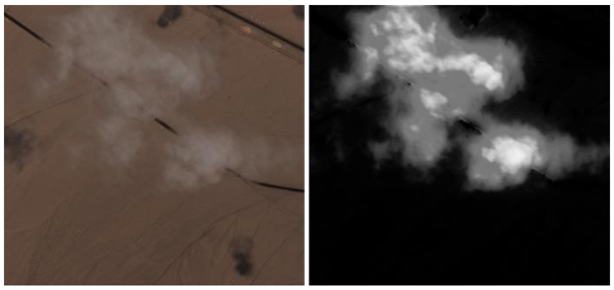


Figure 4-5. Clouds in the left image are detected using the fully trained U-NET model (right)

5. RESULTS – DATA TRANSPORT

In the following sub-sections, we describe PED performance for each of the demonstrations.

Data transport testing was accomplished using a variety of different data types / file sizes as summarized in Table 5-1

Table 5-1. Data types / file sizes for PED demonstrations

Type	# Files	File Size (MB)	Notes
OPIR 2k x 2k	495	160	3 sources, 20 frames per file, 165 files per source
OPIR 4k x 4k	10	640	20 frames per file, 10 files (20 sec)
SAR	1	666	1 frame
EO/IR	4	21.947-30.032	

Data transport (DT) was tested with three primary configurations: AWS, AWS + StarLink, and AWS + Internet. Each test included a Transmitter, Receiver, and Data Forwarder DT component. Latency measurements were recorded separately for each of these DT components. The DT processing functions are depicted in Figure 5-1. In summary, these functions are described as follows:

1. Decomm Time: time to extract the data from the CCSDS formatting
2. Xfer Time (Source to Cloud): time to transfer file from source to receiver in the Commercial cloud
3. OH: Processing time in Transmitter not included in steps 1 & 2
4. Lambda Processing: processing time in lambda function before Data Forwarder is signaled
5. Lambda OH: processing time in lambda function not related to signaling
6. Xfer time (Comm to GovCloud): time to transfer file from commercial to GovCloud
7. Decrypt: time to decrypt file at Forwarder
8. Virus Scan: time to virus scan file at Forwarder
9. Xfer to S3: time to transfer file to final S3 bucket in GovCloud after virus scan is complete
10. Forwarder OH: Processor time in Forwarder not included in steps 6-9 and is negative due to overlap with Receiver

CCSDS decoding (or decommutation) occurs in the ground segment. Although originally, we had had the decommutation step following transport into GovCloud, we found that it was more efficient to do it as part of the pre-processing by the Transmitter function at the ground station. Since for our architecture, ground station processing was required to forward or bundle incoming data into files or other data groupings, it made sense for the decommutation to occur as part of this processing.

A summary of CCSDS decommutation times and rates is shown in Table 5-3 for the various data types. Note that as the file size increases, performance improves, most likely due to fixed overhead in file processing that is diluted as the file size increases. Generally, larger files show a roughly equivalent rate while smaller files suffer a lower rate most likely due to overhead associated with performing processing of a file regardless of size.

Decryption and virus scanning cannot occur until the data reaches the AWS GovCloud; thus, these are activities performed by the Data Forwarder. For each of the data types tested, we were able to achieve a rate of approximately 2,800 Mb/sec. A 160 MB 2k x 2k OPIR data file, then, took approximately 0.5 sec to decrypt. In contrast, the virus scanning rate appeared to be linear with file size – which meant that the time required to scan a 27.4 MB EO/IR file was 1.39 sec whereas it took 1.59 sec for a 160 MB OPIR 2k x 2k file.

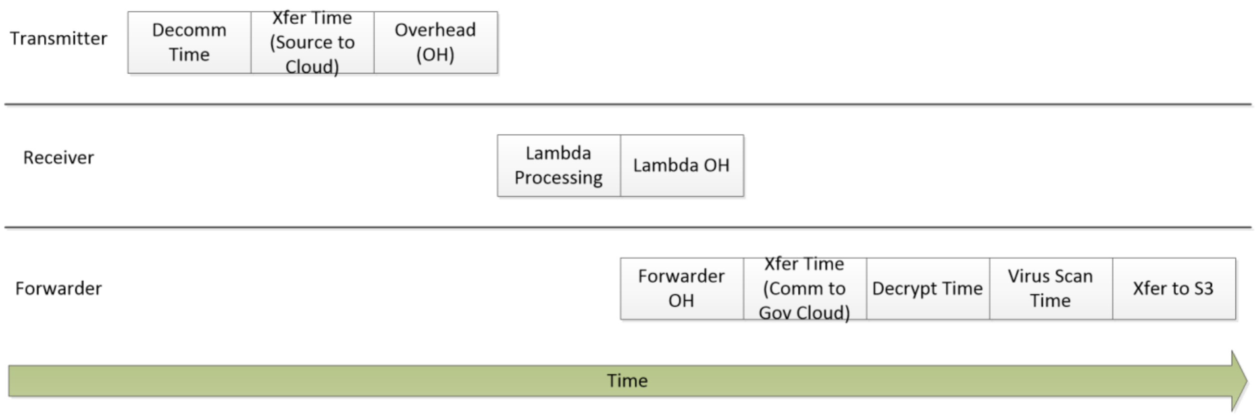


Figure 5-1. DT Processing Steps

Table 5-2. DT Timing Results for Two Typical OPIR Cases

Step	Component	Measurement	casino-4k (6 Jan 2021)		casino1-aft (7 Jan 2021)	
			mean time (s)	End-to-end %	mean time (s)	End-to-end %
1		Decomm Time	6.90	21.54%	1.40	12.92%
2		Xfer Time (Source to Cloud)	7.89	24.63%	2.79	25.74%
3	Transmitter	Transmitter OH	0.44	1.37%	0.12	1.13%
4		Lambda Processing	0.75	2.35%	0.72	6.65%
5	Receiver	Lambda OH	1.09	3.41%	1.10	10.12%
6		Xfer Time (Cloud to Gov Cloud)	3.65	11.40%	0.98	9.07%
7		Decrypt	1.81	5.64%	0.46	4.22%
8		Virus Scan	1.76	5.49%	1.28	11.78%
9		Xfer to S3	7.72	24.09%	1.98	18.24%
10	Forwarder	Forwarder OH	0.03	0.08%	0.01	0.11%

Table 5-3. CCSDS Decommutation Performance

Data Type	File Size (MB)	Decomm Time (s)	Decomm Rate (Mb/s)
EO/IR 967	21.9	0.12176	180
EO/IR 2132	26.9	0.15119	183
EO/IR 2404	30.0	0.15567	196
EO/IR 2428	30.0	0.20009	170
OPIR 2kx2k	160.0	1.61	833
OPIR 4kx4k	640.0	6.9	736
SAR	666.0	6.1	888

End-to-End Timing for OPIR Data Transport

For the 3-sensor 2k x 2k OPIR demonstration, data for one source was transmitted from one AWS region (Ohio) and forwarded to a second region (Oregon). From the Oregon region, data from two additional sources along with the Ohio data were transferred to the Oregon GovCloud, where it was decrypted, virus scanned and made available to the MDP process.

Example End to end DT timing results for OPIR (2k x 2k) are shown in Figure 5-2.

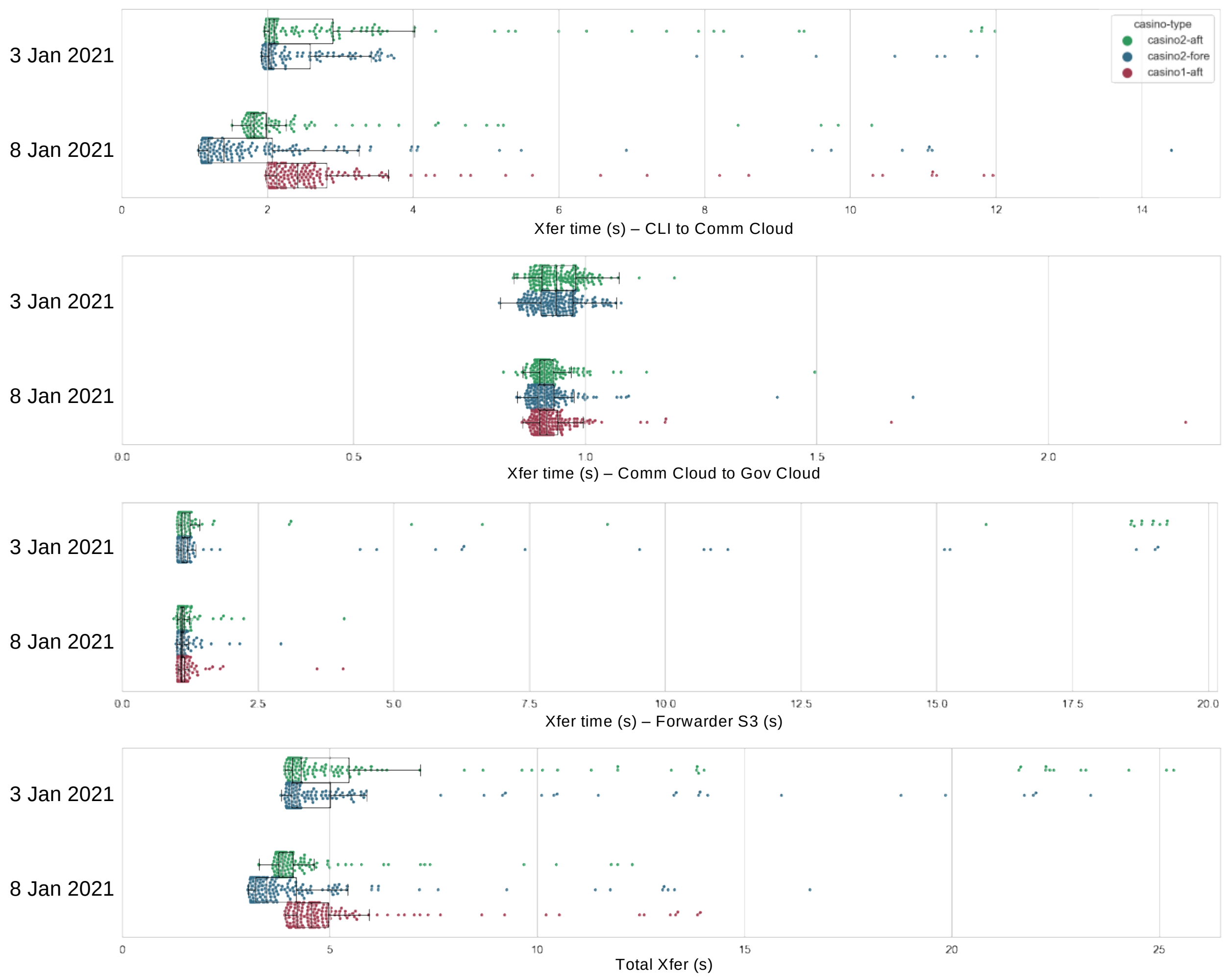


Figure 5-2. End-to-End Timing Results for OPIR Data Transport

End-to-End Timing for SAR Data Transport

The SpaceX StarLink communications channel was used to transfer the single 666 MB SAR test data file from the source ("ground station") to AWS for mission data processing. This testing proved to be important because it uncovered the fact that the original DT design was not robust against unreliable links. In addition to fixing the DT response to dropouts, the resulting design change dramatically improved performance.

The SAR data transport tests were run sourcing the file from a SpaceX laptop connected to a StarLink data terminal in the Los Angeles area. Tests were scheduled during periods when satellite contacts were frequent, but communications dropouts did occur during many, if not all, of these runs. The dropouts were a major source of variation within the tests, but all file transfers were successful in spite of the dropouts.

On average, the times for SAR file transport were:

- Decomm Time: 5.8 sec
- StarLink Xfer: 484.8 sec
- Xfer AWS Commercial to GovCloud: 3.7 sec
- Decrypt Time: 1.9 sec
- Virus Scan Time: 0.8 sec
- Xfer to Forwarder S3: 4.9 sec

EO/IR End-to-End Data Transport

For the EO/IR demonstration, the data were sourced from an AWS ground station in Bahrain. Figure 5-3 shows that the transfer from the transmitter in Bahrain to Mission Data Processing in AWS Oregon is the largest contributor to the total DT processing time. In addition, it has the largest variation.

Pareto analysis for the demo results indicates that improving the Transmitter - Xfr time would result in the greatest improvement in reducing the end to end processing time as it accounts for approximately fifty percent of the time consumed by the DT process for EO/IR files. The virus scan time is second, but has a bimodal feature that suggests the possibility of improving the time taken the majority of the time. Next, a speedup of the lambda function would yield the most improvement.

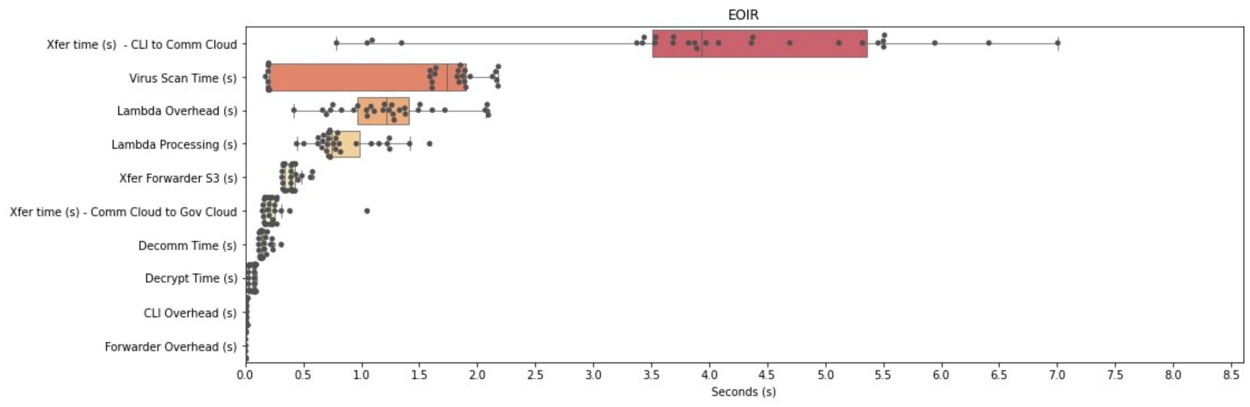


Figure 5-3. Timing and variance for each of the DT functions for the EO/IR demonstration

6. RESULTS – MISSION DATA PROCESSING

The full Mission Data Processing (MDP) architecture for all demonstrations is shown in Figure 6-1

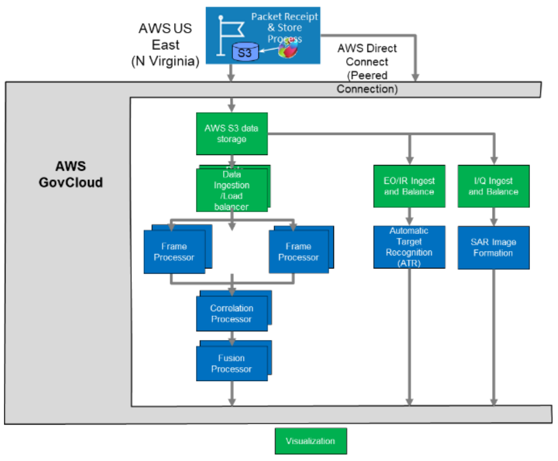


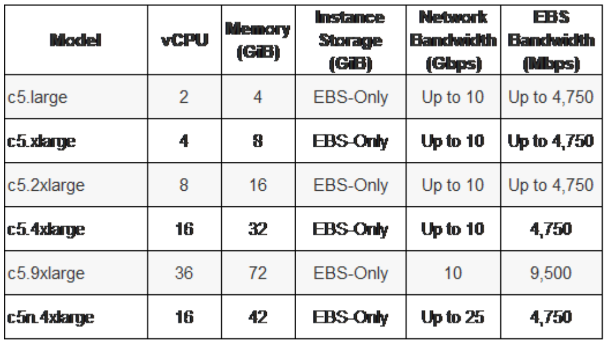
Figure 6-1. Overview of MDP Architecture

MDP Results – 2k x 2k OPIR Demonstration

Six different services were run in the 3 OPIR sensor demonstration – each on an AWS instance that was "spun up" using an Infrastructure as Code (IaC) script. The ***Data Ingest/Playback*** application reads and plays back data frames - A c5.large EC2 instance is used for each sensor data stream. The ***BKG Processor*** ingests raw, calibrated full-frame images, performs clutter suppression, and outputs clutter-suppressed full-frame images. A c5n.4xlarge EC2 instance is provisioned for each sensor data stream. The ***TDE Processor*** ingests the clutter-suppressed frames, performs track-before-detect processing (including full-frame detection – or track initiation, track filtering – or track extension, and signal extraction) and outputs 2D tracklets. TDE runs on a c5.4xlarge instance for each sensor datastream. The ***CORR Processor*** ingests 2D tracklets from multiple sensors, performs multi-sensor measurement correlation, and outputs associated measurements. One c5.9clarge EC2 instance is used for all CORR processing. The ***FUS Processor*** ingests associated measurements, performs state vector estimation, and outputs 3D tracks. It runs on a c5.2xlarge EC2 instance. Finally, the ***Data Analysis/App Dashboard*** applications serve as the user interface for executing and running the MDP in AWS. Each of these applications is run on a c5.large EC2 (Elastic Search) instance.

The 3 sensor OPIR applications are deployed using AWS c5 EC2 instances to take advantage of the higher frequency CPUs available in Compute Optimized instances. The c5 instances contain Intel Scalable Platinum processors with extremely high core counts and base frequencies at or above 3.0 GHz. Additionally, instances with an 'n' (ex. 'c5n.4xlarge') denote higher network bandwidth and are used as background suppression application hosts due to large message sizes of raw sensor frames. The EC2 instance specifications are summarized in Table 6-1.

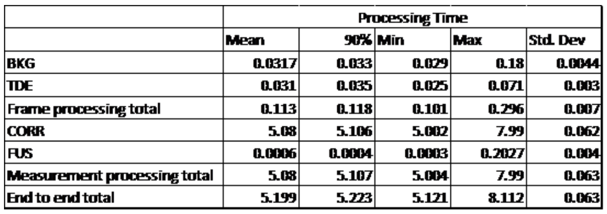
Table 6-1. Summary of AWS EC2 Instances



Model	vCPU	Memory (GiB)	Instance Storage (GiB)	Network Bandwidth (Gbps)	EBS Bandwidth (Mbps)
c5.large	2	4	EBS-Only	Up to 10	Up to 4,750
c5.xlarge	**4**	**8**	**EBS-Only**	**Up to 10**	**Up to 4,750**
c5.2xlarge	8	16	EBS-Only	Up to 10	Up to 4,750
c5.4xlarge	**16**	**32**	**EBS-Only**	**Up to 10**	**4,750**
c5.9xlarge	36	72	EBS-Only	10	9,500
c5n.4xlarge	**16**	**42**	**EBS-Only**	**Up to 25**	**4,750**

The MDP architecture was designed for low-latency, real-time, data intensive processing. The EC2 Instance type and micro-service based architecture ensures the application processing scales to handle increased load from sensor data, while maintaining system performance. The 3 Sensor OPIR Demonstration highlighted horizontal application scaling for data intensive processes. Below, the speed breakdown of each process per 2k x 2k dataframe is summarized in Table 6-2.

Table 6-2. 2k x 2k OPIR Demonstration Processing Latency Summary



	Processing Time				
	Mean	90%	Min	Max	Std. Dev
BKG	0.0317	0.033	0.029	0.18	0.0044
TDE	0.031	0.035	0.025	0.071	0.003
Frame processing total	0.113	0.118	0.101	0.296	0.007
CORR	5.08	5.106	5.002	7.99	0.062
FUS	0.0006	0.0004	0.0003	0.2027	0.004
Measurement processing total	5.08	5.107	5.004	7.99	0.063
End to end total	5.199	5.223	5.121	8.112	0.063

The application processing rate for each individual processing application exceeded the processing rate of the provided data stream (20 Hz/sensor). As can be seen in Table 6-2, BKG, TDE, and FUS performed better than the desired .05 seconds/update needed to stay within the latency rate of the input data stream. Due to some network latencies between playback applications and the Mono-Track applications, the Frame processing total was slightly below the measurement rate. The Correlation application, as designed, includes a 5 second buffer to ensure out of sequence measurements from the multi-sensor platforms can be properly sequenced before fusion processing – this introduces latency, but not a processing bottleneck. Overall, the cloud architecture demonstrated the ability for processing applications to scale to the necessary load of the input data streams and maintain low-latency execution and performance.

MDP Results – 4k x 4k OPIR Sensor Demonstration

The mission data processing pipeline used the same steps as in the 3 Sensor 2k x 2k OPIR Demonstration. Because the data being processed is only from a single sensor instead of 3 sensors providing stereo coverage, this demonstration was meant primarily to test data flow and the background suppression and mono-sensor tracking components of the processing architecture.

The MDP AWS processing for the 4k x 4k Demonstration showed promising performance for application processing and highlighted opportunities of investigation and improvement to ensure low-latency performance in future efforts. The processing applications were able to maintain, on average, the rate of the 10 Hz input 4k data frame. In Figure 6-2, the application processing times of the applications run during the 4k x 4k MDP demonstration are shown (top), including breaking out BKG into component services: Background suppression, Variance Calculation, and Autonomous Multiple Model (AMM). This last AMM service is designed to optimally choose between multiple background suppression algorithms and also includes the elements of BKG that package and transmit the suppressed frame downstream. For this demonstration, we used a single background suppression algorithm, so the AMM service within BKG is simply a message formatter and transmitter.

The application processing speed for the 4k x 4k frame data was sufficient to support the input data requirement of the 10 Hz 4k x 4k pixel data stream (~2.68 Gbps, 16 bit data). Minor improvements in application processing I/O would be desirable to ensure that the data transport within the MDP chain maintains low-latency for the required bitrate. The AMM service within BKG was further analyzed (Figure 6-2, bottom) to break out the specific application processing times of the internal services. The plot shows that the latency within the AMM service was mostly due to the message serialization, which is part of the communication processing within the overall service. The AMM message serialization meant processing speed was slightly higher than the desired 0.1 second per message, and thus drives the overall processing latency of the entire application. Therefore, if we were to reduce the communication latency, the overall processing of the application would comfortably keep up with the desired bitrate of the frame data. In all, the MDP AWS architecture and applications demonstrate the ability to support an increased data load, while maintaining low-latency performance.

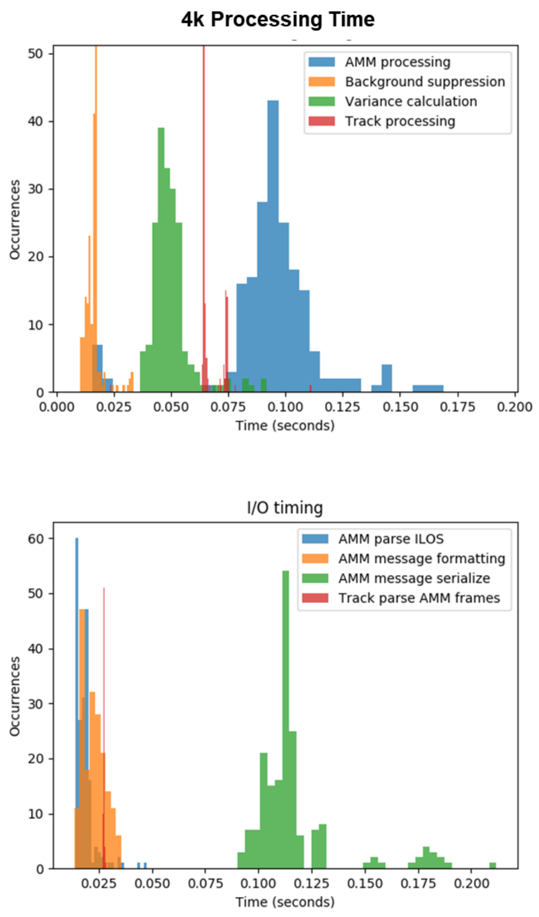


Figure 6-2. 4k x 4k OPIR Application and Communications Processing Times

MDP Results – 30 Sensor 2k x 2k OPIR Demonstration

In order to meet the needs of the objective CASINO constellation and Blackjack demonstration, ground processing must be able to scale as nodes are brought online in the constellation. The 3 Sensor 2k x 2k OPIR Demonstration proved out the data transport and processing architecture that originates with simulated frames and ends with fused 3D tracks; the 30 Sensor OPIR Demonstration showed the dynamic scalability of the processing architecture for a much larger scenario.

The 30 sensor demonstration consisted of 10 contemporaneous copies of the 3 sensor demo, for a total of 30 sensors streaming to the cloud. Because many of the input streams are copies of each other, they result in the same processing and 2D track outputs downstream. This is not as realistic as if a much larger raid were simulated with multiple satellite orbital planes observing the scenario, but this simulation clearly shows how the system performs on a larger scale.

The AWS instances were the same Compute Optimized EC2 instances used in the 3 Sensor OPIR Demonstration, elastically scaled to meet the data load of the 30 sensor demonstration. The Correlation application is not currently horizontally scalable due to limitations associated with the algorithms being used, but is a target of future development at SciTec. The Fusion Processor, on the other hand, *is* horizontally scalable, but was not for this demonstration due to the efficiency of the algorithms employed which obviated the need to scale. A summary of the timing latencies recorded for the 30 sensor demonstration is shown in Table 6-3.

Table 6-3. 30-Sensor OPIR MDP Timing Summary

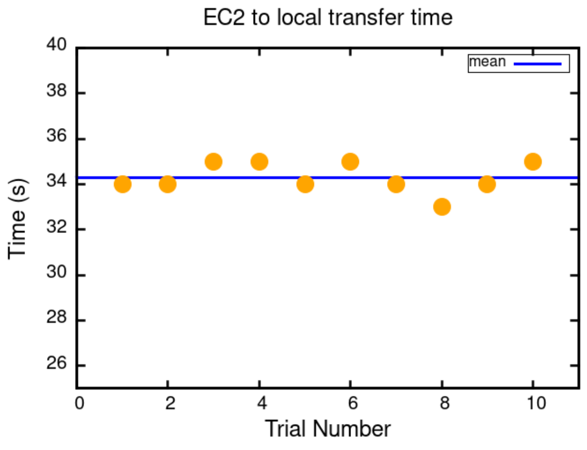
	Processing Time				
	Mean	90%	Min	Max	Std. Dev
BKG	0.0315	0.0324	0.029	0.221	0.005
TDE	0.03	0.034	0.025	0.076	0.0028
Frame processing total	0.108	0.113	0.1	0.333	0.0068
CORR	24.3	59.8	5.008	78.184	21.6
FUS	0.0005	0.0004	0.0003	0.13	0.0033
Measurement processing total	24.27	59.8	5.008	78.184	21.61
End to end total	27.52	62.718	5.125	78.313	22.028

The MDP ability to elastically scale and orchestrate applications was crucial for the 30 Sensor OPIR Demonstration. Increasing the number of instances of background suppression and mono-tracking applications within the AWS framework provided the necessary processing power for the large data influx.

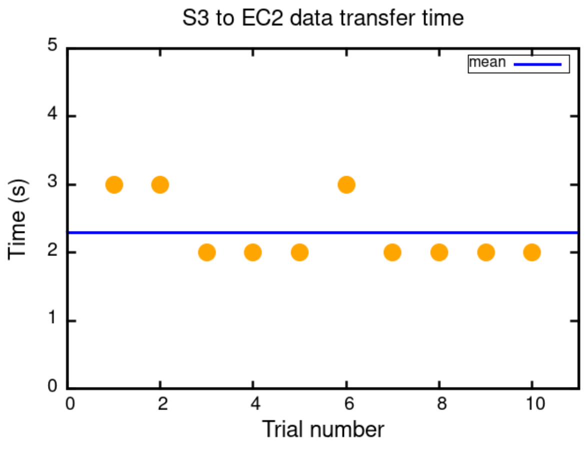
The application processing times for the 30 Sensor OPIR Demonstration were very similar to those of the 3 Sensor OPIR Demonstration. The BKG and TDE applications were scaled horizontally to maintain the processing speeds beyond the required .05 seconds/dataframe. The highly efficient processing in the track Fusion service did not require horizontal scaling and still maintained the processing speeds required for the input data stream. The single CORR processor used to process incoming data from all 30 sensors showed increased processing latency over that observed in the 3 sensor demo due to the 10x data being received. As mentioned in the above sections, we are currently working on an enhancement that will allow CORR to scale horizontally in order to improve full system throughput. The 30 Sensor OPIR Demonstration highlighted the MDP elasticity and orchestration capabilities by maintaining processing speeds even when taxed by increased data loads.

MDP Results – SAR Demonstration

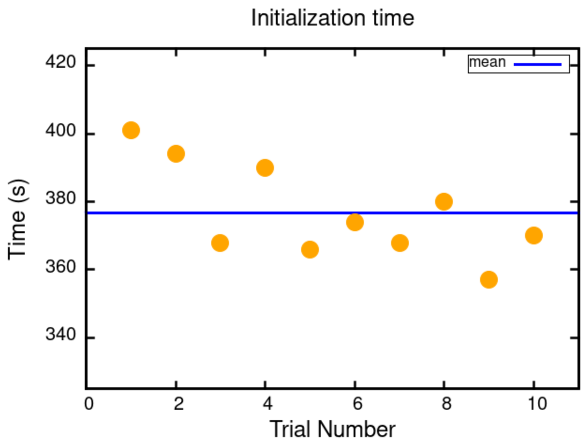
For the SAR demonstration, a c5.9xlarge EC2 instance with 72 GB of memory was used. The timing results for different stages of the SAR processing pipeline are given in Figure 6-3. The results show the timing for (a) the EC2 to local file transfer, (b) the transfer a 670 MB file from S3 to EC2, (c) application initialization, and (d) the time taken to construct a SAR image from raw IQ data. It should be noted that operationally, the application could be initialized once and kept running to avoid lengthy start up times. The raw data file corresponds to high-precision complex data (complex128) with size 8,760 x 8,760 pixels. The final data product that is transported to the local user is a 1.7 MB float32 array of RGB values, hugely reduced from the complex SAR data file. The time required to process the 1x1 meter resolution image shown in Figure 6-3 can be decreased with optimized Fourier transforms.



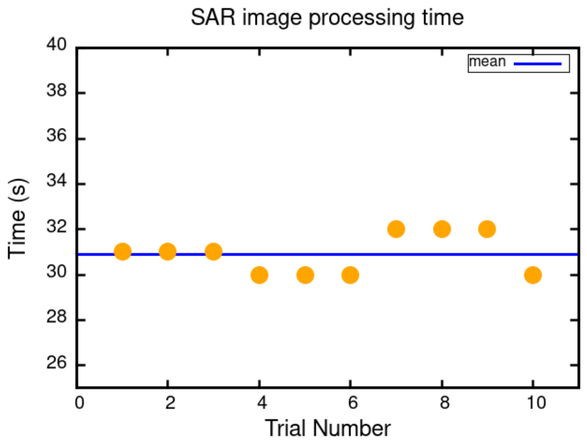
(a)



(b)



(c)



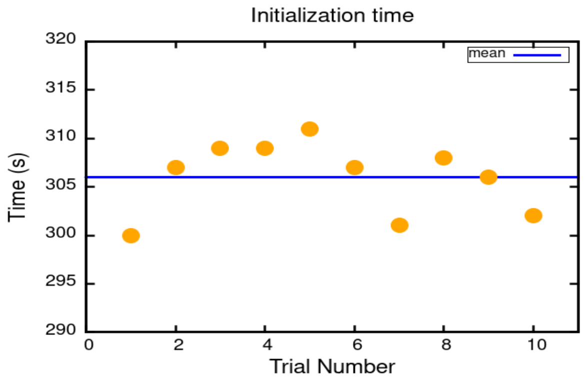
(d)

Figure 6-3. SAR Timing Results. (a) EC2 to local visualizer file transfer, (b) transfer from S3 to EC2, (c) initialization time for the EC2 application, (d) time required by the SAR application algorithm to process one 670 MB IQ image

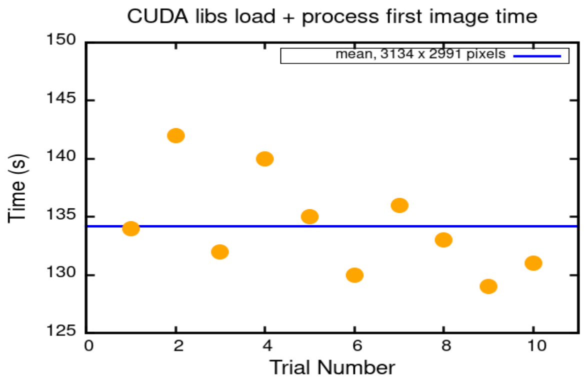
MDP Results – EO/IR Demonstration

For the EO/IR demonstration, we employed a GPU-enabled g4dn.xlarge instance, which has 16 GB of memory. Based on a series of data processing trials, the maximum usage of this instance peaked at ~11%. For the file sizes tested in the demo (23 to 31.5 MB), the file transfer times were not sensitive to file size and averaged ~1 sec for transfer between the S3 bucket and EC2 and ~7.8 sec for transfer between EC2 and a local workstation.

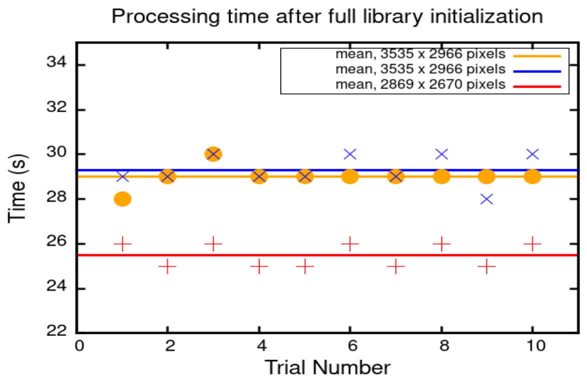
Although the original images were slightly different sizes, the output product that is transferred over zmq is the same for each input image, and is displayed as a 7,200 x 1,800 4-panel picture, each with target recognition or cloud masking results clearly and separately identified. In Figure 6-4, timing is plotted for 10 trials of the EO/IR processing demonstration.



(a)



(b)



(c)

Figure 6-4. EO/IR Processing Initialization Times

The times shown in Figure 6-4 (a) include EC2 instance creation, docker initialization, and launching docker. The times shown in (b) include loading the CUDA libraries and processing the first image. Figure 6-4 (c) shows the processing times for the images processed after library initialization. From the image processing statistics, it was shown that, if linear, the processing time scales with slope greater than one with increasing image area.

7. CONCLUSIONS AND OPERATIONAL IMPLICATIONS

Over the course of the DT prototype development, the DT transport and processing components were optimized to provide a streamlined, low-latency pipeline. The initial focus of the prototype was to assess the feasibility of pushing high-rate full-frame OPIR data sets from a ground station location to a secure cloud processing location. A high end goal for the prototype was to determine if data could be streamed in real-time from multiple receipt locations into a secure cloud processing center and then processed into actionable messages disseminated to field assets in a timely manner.

Operational Implications – Data Transport

For the prototype effort the DT was set up to emulate a transfer process that could be used to transfer data across the AWS cross-domain diode into a classified processing region. This emulation necessitated the use of an S3 file-based transfer between the commercial and 'secure' processing regions. Moving data in files and storing the files into an S3 storage unit introduces latency that could likely be reduced through the use of streaming mechanisms. Streaming is a potential future option on the AWS road-map for cross-domain transfers.

To reduce latency, DT components can be scaled to perform parallel processing. However, maintaining order and synchronization of files flowing into the MDP engines is also required. During development of the prototype, it was apparent that adding parallel operations results in increasing the chances of having files show up out of order, which results in introducing additional latency to re-order the files prior to pushing the data into the MDP engines. Introducing a level of 'store and forward' to buffer, and re-order data prior to transferring the data into the MDP elements is necessary to support maintaining time ordering.

DT prototype development successfully showed the ability to push data across the commercial networks and into a secure processing area. Transfer of full frame data is likely to occur for operations associated with validation and/or calibration of on-orbit processing. For non-real time transport of full-frame data, the 'store and forward' buffering approach can be used effectively.

In an operational version, on-orbit assets are likely to process the raw sensor frames into intermediate products. Transferring the intermediate products from multiple satellites to a secure cloud processing region where the data can be correlated and fused into tactical data and disseminated to field assets is well within the DT capabilities. DT is built and has been demonstrated to support many different types of data files.

Operational Implications – Mission Data Processing

We demonstrated the ability of three of the four mission processing services in our OPIR MDP – background suppression, mono-tracking, and fusion – to horizontally scale to maintain real-time OPIR performance processing of data from up to 30 sensors simultaneously. To operationalize the elastic processing architecture, we will need to implement horizontal scalability for the one remaining service - correlation. The re-factoring required to do this is actively being worked.

One of the most striking findings of this effort, perhaps, is the very high latency associated with "spinning up" instances for processing, as well as for transferring data (either raw data or processed results) from an AWS instance to a local machine for visualization, analysis, or further downstream processing. For example, initializing the SAR and EO/IR EC2 instances and applications took > 6 minutes. For an operational system, it is likely that the savings associated with only turning on instances when needed cannot be realized due to the latency associated with initialization – i.e., these instances would have to be always on, or we would need to establish alternate approaches to dynamically initializing services in AWS. This is an area for further research and prototyping.

In each of the CASINO PED mission data processing demonstrations, we executed IaC scripts that spin up the required compute services, handle data transfer between AWS storage and compute services, process the Mission Data on that compute service, and finally transfer the data to a local workstation for visualization. Currently, these IaC scripts are launched by a user each time processing is desired. For an operational PED system, processing would be orchestrated automatically, using metadata attached to the mission data to determine what processing chain to use, its configurations, and the resources required to execute the processing.

Currently, the PED Prototype supports full-frame processing and multi-sensor correlation and fusion to output 3D tracks from input OPIR sensor data, as well as SAR data pre-processing and visualization and EO/IR automatic target recognition and cloud masking. Although our prototype demonstrates versatility and multi-mission utility, there is still work to be done in terms of making the PED prototype compatible with current operational OPIR systems as well as potential future CASINO assets. For example, the OPIR microservices currently expect full frame data and output mono-sensor 2D tracklets, which are then correlated and fused to generate 3D tracks. However, in bandwidth constrained environments, OPIR systems frequently do not disseminate full frame data and instead produce exceedances / rep returns, which are then provided to downstream processes for correlation and fusion. Although fundamentally, our correlation and fusion engines can support processing rep returns vs. mono 2D tracklets, some changes would be needed in the PED prototype to support this type of processing if required for targeted systems.

References

1. Sune R.J. Axelsson, "Fast simulation of SAR raw data from complex scenes," Proc. SPIE 4543, SAR Image Analysis, Modeling, and Techniques IV, 2002.
2. Skolnik, Merrill I. Radar Handbook. McGraw-Hill 2008.
3. Carrara, Walter G., et al. Spotlight Synthetic Aperture Radar: Signal Processing Algorithms. Artech House Inc, 1995.
4. Lin, T-Y., et al., "Focal Loss for Dense Object Detection," arXiv:1708.02002v2, 7 Feb 2018
5. Ronneberger, O., et al., "U-Net: Convolutional Networks for Biomedical Image Segmentation," arXiv:1505.04597v1, 18 May 2015
6. Lam, D., et al., "xView: Objects in context in Overhead Imagery," arXiv:1802.07856v1, 22 Feb 2018.
7. https://www.planet.com/products/hi-res-monitoring/