35th Space Symposium, Technical Track, Colorado Springs, Colorado, United States of America
Presented on April 8, 2019

AN INNOVATIVE, NEAR-REAL-TIME APPROACH TO SPACE SITUATIONAL AWARENESS (SSA) DATA SIMULATION

Susan P. Hagerty
Ball Aerospace, shagerty@ball.com

Khurrum Ansari
Ball Aerospace, kansari@ball.com

ABSTRACT

Space Situational Awareness (SSA) is vital to maintaining our Space Superiority. We present an innovative, high fidelity, time-based simulation tool, RT-PROXOR™ (Real Time Proximity Operations and Rendering), which supports SSA by generating realistic mission scenarios including sensor frame data with corresponding truth. This is a unique and critical tool for supporting mission architecture studies, new capability (algorithm) development, current/future capability performance analysis, and mission performance prediction. RT-PROXOR™ provides a flexible architecture for sensor and resident space object (RSO) orbital motion and attitude control that simulates SSA and rendezvous scenarios. The major elements of interest are based on the ability to accurately simulate all aspects of the RSO model, viewing geometry, imaging optics, sensor detector, and environmental conditions. These capabilities enhance the realism of mission scenario models and generated mission image data. This paper advances the work presented in our 2017 PROXOR™ paper, and presents our new RT-PROXOR™ capabilities, current execution time performance results, and path forward.

RT-PROXOR™ is designed as a component in a software-only or Hardware-In-The-Loop (HWIL) simulation. In this latest version, much of the code is resident on GPUs (industry COTS graphical processing units), thus greatly increasing execution speed. RT-PROXOR™ uses mission scenario information to set up the scenario to be modeled, and then each frame it is given the current pointing information and clocking and then generates the image using high fidelity optics and detector models. Given the pointing angles and orientation for the current frame, we calculate the changing solar and Earth illumination angles of the satellite. The synthetic satellite image is rendered at high resolution and aggregated to the focal plane resolution resulting in accurate radiometry even when the RSO is a point source. Critical aspects such as intra-frame smear and a high-fidelity detector model are included. In this paper we present an overview of RT-PROXOR™ plus simulation results (generated images) for SSA missions.

INTRODUCTION

The US is increasingly dependent on its space assets for commercial, civil, DoD, and intelligence community use. For example, warfighters are extremely reliant on GPS for support of surgical strikes. Adversaries see our ever-growing dependence on space assets as a critical vulnerability to be exploited and as they look to counter US superiority in space, it will become the next battlefield. The US government has a need to catalog all objects in space, characterize them, detect uncorrelated targets and identify threats/risks to our space assets. To meet those needs, the Air Force and Intelligence Community continue to develop space-based and ground-based systems for determining and exploiting metric (location) information and resolved and non-resolved signatures of RSOs and targets, with the long term objective of identifying objects and deriving quantitative and functional information about them[1].

Threats from foreign nations are becoming more prevalent. Plus the sheer number of objects in space (upwards of 22,000 objects are being tracked with >= 10 cm size and about 500,000 objects are present in space

from 1-10cm size)* makes it increasingly difficult to maintain custody of these objects and difficult to ensure we can find new, potentially adversarial objects, detect changes in objects, and identify nefarious behavior in a timely manner. The growth in trackable objects is shown in **Exhibit 1**. The threats to our space assets in GEO are of particular concern, and additional space-based SSA platforms need to be fielded to counter this threat. A notional space-based observer searching the GEO belt is shown in **Exhibit 2**. There is a need to develop algorithms and perform optical system design trades in support of SSA to solve uncorrelated track (UCT) challenges: resolving conjunctions, detecting new targets, and detecting changes in target characteristics[2]. This paper covers the Ball RT-PROXOR™ simulation tool developed to support solutions to fill these critical gaps and key shortfalls in space surveillance capabilities. To support these shortfalls, there is a need to develop and field robust algorithms, which must be tested over a wide array of mission scenarions, including modeling & simulation of long scenarions, e.g., a Day In The Life, that have high fidelity for confidence building. Testing needs to be done for both open loop and closed loop systems. The focus of RT-PROXOR™ is on fast simulation to support end-to-end software and hardware testing in an efficient manner, allowing for hundreds of long-duration mission scenarios to be run, ensuring robust algorithms are developed early in the development efforts.

Performing a wide spectrum of tasks from SSA algorithm development including implementation and testing, to SSA characterization and mission-level analysis with varying degrees of fidelity requires multiple tools with differing objectives. We have previously developed a tool called PROXOR™, which Ball uses to produce very high fidelity images over time[34]. PROXOR™ is an excellent tool for performing analyses that do not require close to real-time performance, and hence its full high fidelity capabilities can be brought to bear. This is an extremely valuable tool for accurately representing true satellite images for purposes of characterization, and accurate performance estimation over short time periods. On the other extreme, there is a need for real-time or near-real-time scene generation that can generate minutes to hours of mission data (scenes). This capability is required for robust testing of algorithms over time, and also for mission analyses / Day-In-The-Life (DITL) testing. We are currently developing this tool, called RT-PROXOR™, which is a critical component in our end-to-end Software simulation and High-performing Algorithm Development Environment (SHADE). This is primarily designed for closed loop processing, but is equally applicable to open loop processing (generation of “datacubes”, i.e., images over time) scenarios. This paper provides a snapshot in time (our Spiral 5 version) of our current RT- PROXOR™ development activities, while also providing a brief overview of Ball’s mature PROXOR™ tool.

RT-PROXOR™ currently simulates the observer(s) platform and sensor pointing, trajectory(ies), and sensors. We include high fidelity detector models, artifacts, RSO trajectories, star fields (including viewing high star count areas such as the galactic core), and environmental effects (uniform background and radiation hits). For RT-PROXOR, the sensors are assumed to be space-based. RT-PROXOR™ is radiometrically accurate (for Lambertian spheres), and generates realistic high fidelity images over time that are used, for example, to i) assess and compare potential mission architectures (e.g., GEO constellations vs. GEO/LEO constellations for SSA or Missile Warning), and ii) provide image frames over time in RT/Near-real-time (NRT) to support SSA algorithm development and testing and to verify algorithm performance (e.g., Pd vs. SNR/visual magnitude performance, metric accuracy, etc.), The primary objective of RT-PROXOR™ is to provide realistic data for development and test so that SSA algorithms can work well in the real world “right out of the chute”. This in turn improves the probability of successful mission execution, whether the mission is to find threats as part of Indications & Warning (I&W), avoid collisions, perform RSO characterization, or execute other Space Protection missions.

* See, e.g., http://www.bbc.co.uk/news/science-environment-14763668

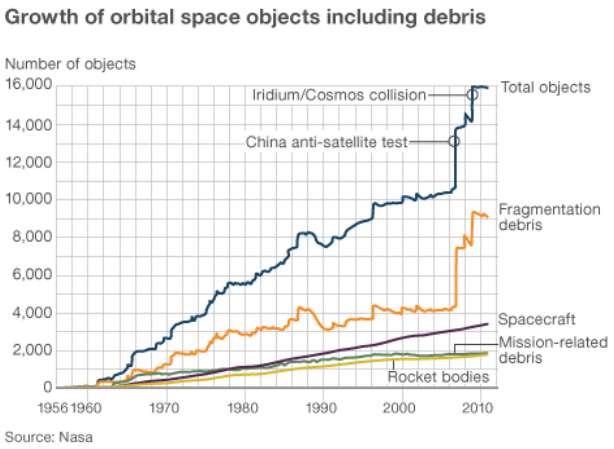

Exhibit 1. The growth of orbital objects has increased significantly over the last 50 years with large spikes due to debris caused by intentional (Chinese ASAT test) and unintentional collisions (Iridium/Cosmos).

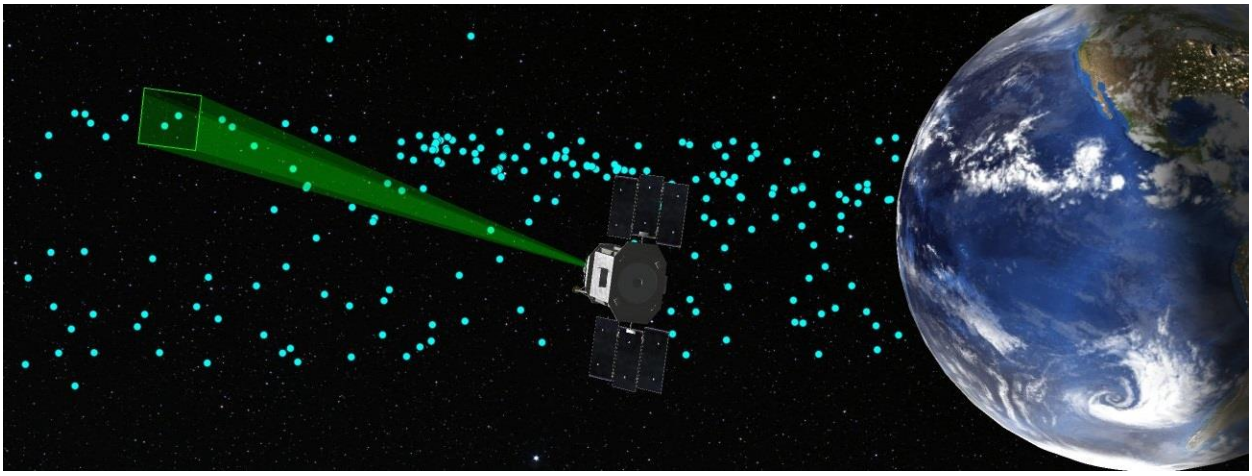

Exhibit 2. Example mission scenario showing geometry of observer in Search mode viewing RSOs and stars in its Field of Regard (FOR).[†]

PROXOR™/RT-PROXOR™ OVERVIEW

Ball's PROXOR™ tool, shown in **Exhibit 3**, provides a **high-fidelity** and **innovative** simulation architecture for modeling the orbital motion, attitude control, radiometry and focal plane images of a sensor or multiple sensors and multiple RSOs in realistic SSA, rendezvous, and proximity operation scenarios. PROXOR™ **enables** the simulation of real SSA mission scenarios, including resolved and unresolved space objects, using actual-validated satellite data (e.g.., real DMSP and Intelsat spacecraft surface properties). PROXOR™ has a Graphical User Interface that facilitates setting up mission scenario and sensor modeling parameters, as well as selecting satellite(s) of interest from an expandable library of satellite models. Ball's 2017 Space Symposium paper (referenced above) contains more information about the advanced capabilities of the full PROXOR™ version.

Ball has developed RT-PROXOR™ to address a critical need for supporting NRT/RT scene generation for Software In the Loop (SWIL) and Hardware In the Loop (HWIL) applications, and is therefore focused on both execution speed and fidelity – and finding the right balance between the two. Required fidelity in many cases can be maintained at much faster speeds than the full PROXOR™. Depending on the exact nature of the required imagery / goals of the analysis, fidelity knobs can be "turned", to greatly increase execution speed, while still maintaining the level of fidelity required. For example, if the user wants to include RT- PROXOR™ in a HWIL

[†] Note that high fidelity PROXOR™ images are shown below in Exhibits 10, 12, and 13.

simulation and the frame-to-frame motion is not too large (~< 0.1 pixels/frame), then intra-frame smear can be modeled at reasonable fidelity without generating the full baseline intraframe modeling, which internally renders hundreds of frames for every FPA frame. This would greatly reduce execution time, while not sacrificing the ability to generate realistic data. ***The ability of RT-PROXOR™ to generate imagery fast (NRT/RT) over long mission periods with good fidelity is invaluable in i) evaluating and predicting mission performance as part of a larger end-to-end mission-level analysis tool; and ii) developing, testing, and verifying successful algorithms for SSA including RSO detection, tracking, and characterization for all objects (including high interest objects), leading to increased confidence in mission success***. In general, RT-PROXOR™ is a subset of the full PROXOR™. The key differences between PROXOR™ and RT-PROXOR™ for space-based EO imaging are i) RT-PROXOR™ has significantly reduced execution speed (up to a factor of 300 improvement), ii) RT-PROXOR™ currently models all objects as Lambertian spheres and therefore does not yet model extended objects using our Satellite library[‡], iii) RT-PROXOR™ can use either full multi-kernel rendering of stars (which PROXOR™ also does) or a fixed kernel, which is much faster and appropriate if the scenario does not have significant clocking or other nonlinear motion, iv) RT-PROXOR™ generally assumes pointing is external[§] since its primary purpose is to be part of an end-to-end Software simulation or HWIL simulation, though it also does perfect pointing to a designated target satellite if configured to do so, and v) RT-PROXOR™ does not include Zernike coefficients for modeling optical aberrations, or second order detector effects.

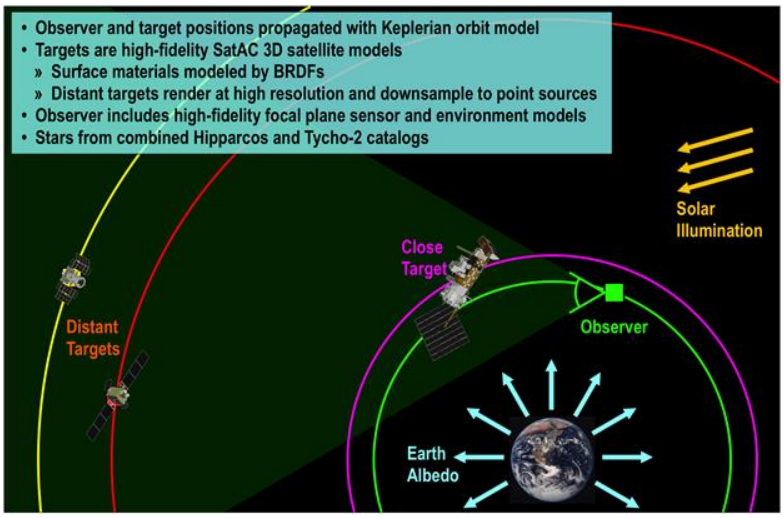

Exhibit 3. PROXOR™ Modeling & Simulation tool is used to model mission scenarios, support SSA algorithm development and verification, support analyses using generated data and SSA algorithms that inform optical system design trades [focal plane Field of View (FOV) / Instantaneous FOV, aperture size, PSF, noise, etc.]

The focus of RT-PROXOR™ is on space-based scenarios, where it i) independently propagates the positions of the observer and one or more target satellites using TLEs from the industry-standard SGP4 propagator, or ii) uses ECI coordinates (encompassing the start of integration through end of integration period motion for each satellite) provided by the external Software Simulation or HWIL interface. The viewing and illumination angles for each target are calculated as a function of time, and 2-D images of each target are rendered for each frame.

RT-PROXOR™ focuses on visible wavelength imagery, but can also render at select short, medium, and long infrared wavelengths. The sensor model is used to determine the sampling of the targets based on pixel IFOV

[‡] This capability will be added in the near future.

[§] RT- PROXOR™ does autonomously propagate the intra-frame motion of the orbits using the SGP4 propagator.

(which is oversampled) and the range to the target. RT-PROXOR™ automatically includes additional oversampling if a minimum number of points are not placed across the target. After high-resolution rendering, the target image is aggregated back to the sensor model sampling. This imaging approach guarantees correct spatial phasing and accurate radiometry even for point source targets. RT-PROXOR™ uses an analytic Lambertian sphere renderer to enable efficient modeling of target satellites, which are generally adequate for SWIL/HWIL applications, where speed is of the essence. **Exhibit 4** shows the RT-PROXOR™ functional block diagram.

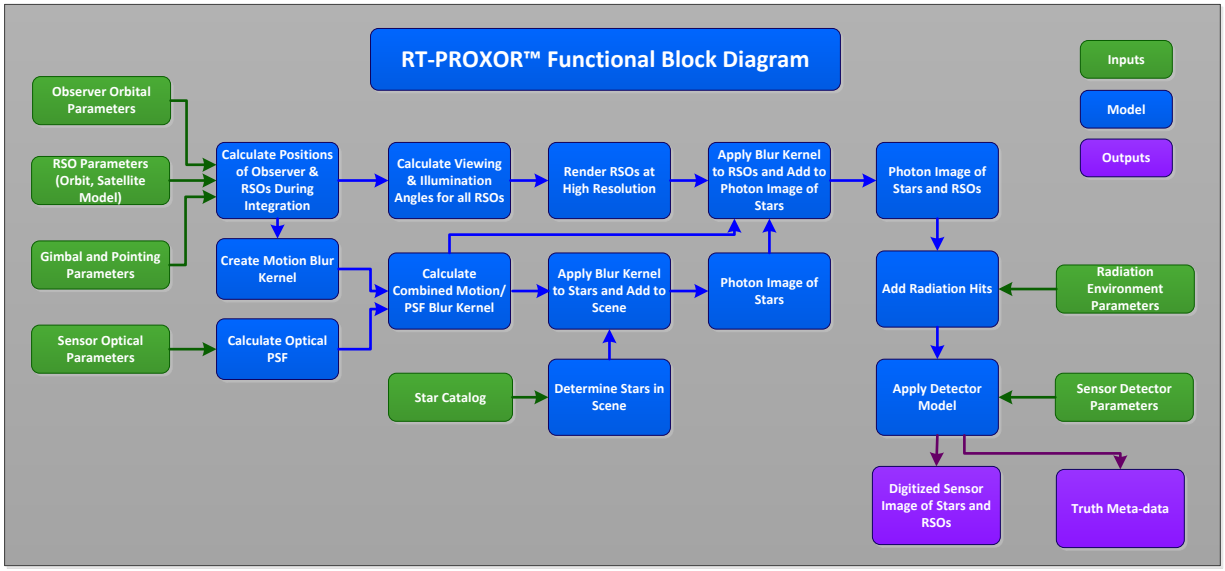

Exhibit 4: The RT-PROXOR™ Functional Block Diagram shows the production of a single image frame of digitized sensor data.

PATH TO RT-PROXOR™

Our approach to generating a RT/NRT tool for scene generation is to leverage key parts of our high fidelity PROXOR™ tool, while realizing signficant speed improvements, which allows for more robust testing and reduces implementation time. In addition, some verification and validation work has already been conducted on the full PROXOR™, which allows us to shorten the testing on RT- PROXOR™ by directly comparing outputs of PROXOR™ and RT-PROXOR™. Speed improvements are primarily realized by moving functionality to the GPU(s). GPU processing cores can perform very repetitive and parallelizable processing tasks at high speed. This applies to many critical 'long pole' items such as star rendering, where an optical model is generated along with a line of sight estimation for each star in the sensors field of view over the course of the integration period. The implementation path from PROXOR™ to RT- PROXOR™ is shown in Exhibit 5.

Software Code Base Conversion	Convert core Matlab functionality into C++ based code for Graphical Processor Unit (GPU) execution.
Updated Pointing Mechanisms	Internal pointing and attitude references were updated from DCM (direction cosine matrix) based expressions to purely 4 element quaternions.
Orbital Propagation	Upgraded the internal orbital propagator to use the industry standard SGP4. SGP4 describes all tracked space objects by the US government using a two-line element (TLE) vector database – facilitates easy modeling of satellites of interest for any date/time.
Hardware in the Loop (HWIL) Interface	-A packetized frame by frame interface for injecting target/acquisition satellites was formulated to allow scenarios/maneuvers in real-time that

	cannot be described by SGP4 TLEs. -This expands upon the capabilities of specialized use cases of rendezvous/acquisition scenarios. -Allows the updating of system parameters (focal plane/field of view/integration period) in real-time to switch a platform's operational mode. This interface can be injected in a purely software controlled interface as well, with the frame by frame updates ingested via a pregenerated JSON file.

Exhibit 5: Major RT- PROXOR™ development steps are on a direct path towards real-time execution

Two important goals of the development are to i) build an end-to-end Software simulation for very early development that potentially provides faster-than-real-time execution, and ii) build the RT-PROXOR™ real-time HWIL system so that it could test complete image processing / tracking systems in hardware early in the program, operating at full speed. Typically, the goal here is 4 – 30 frames per second. Benefits of software simulation using Ball's SHADE tool and custom HWIL that leverages previous Ball efforts are shown in Exhibit 6.

Framework	Benefits
Software Simulation using Ball's SHADE tool	-Available very early in the design process, before HW Engineering Models (EMs) are available -Increases execution speed and potentially can run faster than real-time, so that, e.g., missions such as Day-In-The-Life can be tested in much less than a day of real time -Can easily make updates and quickly re-evaluate performance in Ball's flexibile SHADE architecture
Custom Hardware in the Loop (HWIL), leveraging existing Ball HWIL solutions	-Available early in the design process, before final hardware is available -Runs in real-time on embedded, automated processor in HWIL architecture -Can test and find issues with the actual hardware interfaces, prototype firmware/FSW code early in the program before implementation and test of the final system occurs -Increases execution speed and runs embedded hardware in real-time, allowing hardware interfaces, algorithm performance, throughput and latency to be evaluated and iterated upon to enhance performance -Can easily make updates and quickly re-evaluate performance with Ball's flexibile HWIL architecture
Common to both SW and HW Sim	-Uses RT- PROXOR™ Scene Generator to rapidly generate images with medium-to-high fidelity -Can run many hours of testing over many varied mission scenarios to increase algorithm quality and robustness -Fully automated -Reduces overall schedule by retiring risks early in the program, rather than later when any changes require much more schedule time -Increases program affordability by finding issues and fixing them early in the program when required labor is much less costly -Supports both open loop and closed loop testing -Common scenario specification, top level interfaces, visualization and analysis tools are used in the SHADE Software Sim and our corresponding HWIL Sim, making it very efficient to move back and forth fromSoftware Simulation to/from HWIL -Both approaches use RT-PROXOR and can operate via data cubes or by a packetized frame by frame interface; the packetized interface supports:

	--per frame pointing based on algorithm desired pointing and gimbal modeling --injecting target/acquisition satellites to allow scenarios/maneuvers in real-time that cannot be described by SGP4 TLEs.-This expands upon the capabilities of specialized use cases of rendezvous/acquisition scenarios, allowing us to better model/address threats. -Allows the updating of system parameters (focal plane/field of view/integration period) in real-time to switch a platform's operational mode.

Exhibit 6: Benefits of Software and HWIL Simulation

Software Simulation

Block diagrams of SHADE are shown in Exhibits 7 and 8. In addition to performing modeling & simulation for open loop sidereal and rate track Space Situational Awareness missions, Ball simulates closed loop rate tracking on resident space objects and other objects of interest. SHADE includes high fidelity models for image generation (via RT-PROXOR™), the gimbal/plant, and spacecraft and target dynamics as well as algorithms for image processing, tracking, and control (for command generation). SHADE can be run closed loop (Exhibit 7) or open loop (**Error! R eference source not found.**8). RT-PROXOR™ provides truth so that the performance of the mission image/track processing functions can be evaluated and optimized under realistic conditions. This is a crucial piece necessary for performing realistic mission data analysis for single observers, as well as constellations. In closed loop mode, RT-PROXOR™ ingests the Line-of-Sight (LOS) pointing from the control system and generates the high fidelity image. This image contains the stars and RSOs in the field of view of the observer and is fed to the image processing algorithms. The image processing algorithms include artifact correction, background removal, detection, location, and initial discrimination. The tracking algorithms include stabilization (coordinate transformations to inertial space), tracking, and target of interest (TOI) identification. The control algorithms include compensators to ensure closed loop stability and optimized system performance. Closed loop tracking has a number of advantages to open loop. It concentrates the target energy, improving the SNR and centroid accuracy, hence allowing detection of dimmer targets, and improved tracking performance. It also provides the ability to track the target over large fields of regard. Closed loop tracking can be initiated via a single cue from the ground, or via autonomous acquisition. SHADE also has an Observations (Obs) generator for bypassing the image generation and image processing for rapid turn-around and parallel tracking algorithm development and testing. Ball's SHADE helps reduce risk for future missions by characterizing the ability to robustly track space-based targets, and to optimize systems for increased metric accuracy, leading to improved orbit determination (OD) for TOIs and improved anomaly detection and therefore provides enhanced Indications & Warnings (I&W).

HWIL Simulation

A block diagram of our HWIL (also called Payload-in-the-loop (PITL)) architecture is shown in Exhibit 9. This uses a similar functional architecture to SHADE, but uses an Engineering Model (real hardware processor) for the processing algorithms, and appropriate hardware interfaces from the RT-PROXOR™ Scene Generator/Scene Injector to the processing board, and from the processing board outputs to the Gimbal and Spacecraft Bus models. HWIL testing provides the first opportunity in the program to do thorough testing on real hardware, greatly reducing cost, schedule, and technical risk.

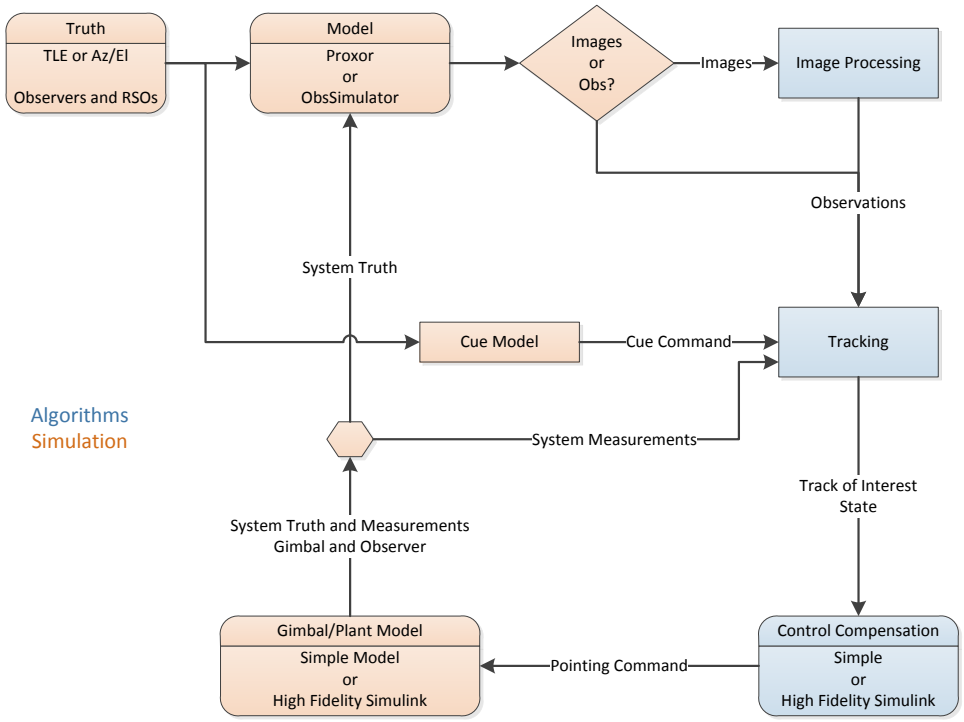

Exhibit 7: SHADE Closed Loop Block Diagram

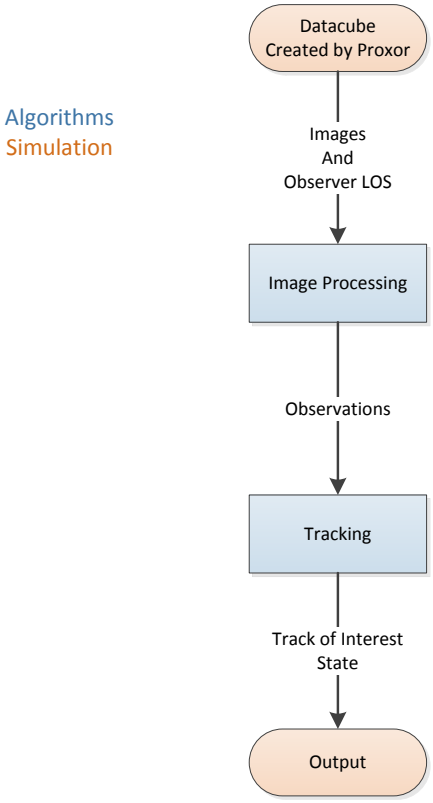

Exhibit 8: SHADE Open Loop Block Diagram

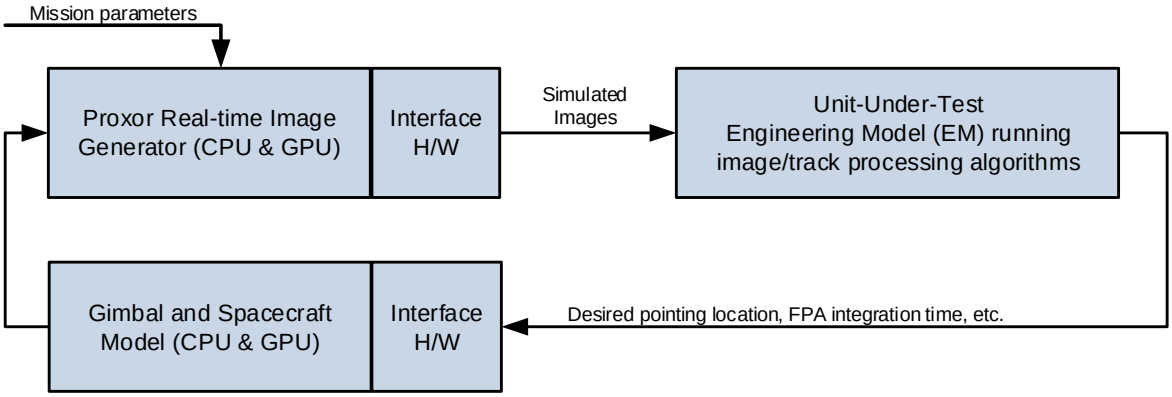

Exhibit 9: RT-PROXOR™ used for Hardware-in-the-Loop testing

RT-PROXOR™ processing

There are multiple steps to simulate an image. Prior to image generation, a mission CONOPS is developed and a set of parameters describing the scenario are generated and exported to a javascript notation file (JSON). This file contains all configurable elements of RT-PROXOR™, describing the target satellites of interest, the acquisition (observer) satellite, optical characteristics of the system, the detector modeling, and mission characteristics (start/stop times, orbits, ranges to targets, etc.). Once the mission parameters are generated, RT-PROXOR™ can start generating the simulated frames. There are five main steps entailed in generating each RT-PROXOR™ frame.

Generate the Position and Attitude of the Acquisition Spacecraft/Gimbal and All Other Satellites of Interest.

The TLE data of the acquisition satellite as well as all target satellites is propagated (determined by the temporal oversampling factor, an input parameter) many times during the integration period to simulate intra-frame smearing.

These parameters can be overridden or added upon via the HWIL interface. This mechanism would facilitate the ability of a satellite to perform a maneuver outside the bounds of its predefined orbital path.

Specialized tracking modes can be developed by updating the pointing via HWIL interface (such as perfectly pointing towards a satellite of interest or a particular point of interest in the star catalog).

The gimbal position data has jitter components (both high and low frequency) that can be added to the pointing data.

Determine All Stars That are Visible on the Focal Plane

Generate a list of culled stars spanning the gimbal pointing throughout its integration period.

The full star list used for this effort is a 2.56 million star catalog.

Create the Background Image of All the Stars onto the Frame Scene

If the integration time of the focal plane is configured to be long, and the gimbal system is moving during this time period, there may be signficant streaking/clocking on each star.

Stars may end up looking stationary, streaking in a direction, or arc shaped depending on the line of sight motion.

The optical model characteristics (the point spread function of the optics) are applied to each star during this

phase.

This is generally the most time consuming module due to the large number of stars that could be on the focal plane, and treating each star individualy to account for motion as translated onto the focal plane.

Render the Target Satellites

Currently, RT-PROXOR™ uses Lambertian spheres to model point source satellites of interest. The full Matlab version has the capability to use in-house developed rendering as well as the Satellite Assessment Center renderer to simulate extended targets as they approach closer on the focal plane.

The earth umbra as well as sun angles are used for Lambertian sphere illumination. If the target is too small, it is first upsampled to accurately model illumination effects due to the light source geometry.

Smearing due to the line of sight changing throughout the integration period as well as optical modelling effects are applied to each target analagous to the star rendering phase.

Add the effects of the High-Fidelity Detector Modeling

This module applies the final parameters of the acquisition system to complete the generation of the image.

This includes effects such as instrument background noise, quantum efficiency, dark current, shot noise, randomly generated spurious radiation tracks, read noise, and various types of quantization models/effects.

Exhibit 10. Steps to Generate a Frame of Data

PROCESSING ARCHITECTURE

A multi-threaded architecture was designed and implemented correlating to logical divisions in the processing as outlined in Exhibit 4 above. Furthermore, it is conceivable that each thread could have its own dedicated GPU, as our approach is designed to take full advantage of multiple GPUs. To this point, we have only used a single GPU resource but there is flexibility in the architecture to expand upon this, which has the potential to further reduce per-frame execution time. Exhibit 11 outlines the software architecture, and Exhibit 12 gives a top level description of the “roles and responsibilities” of each thread.

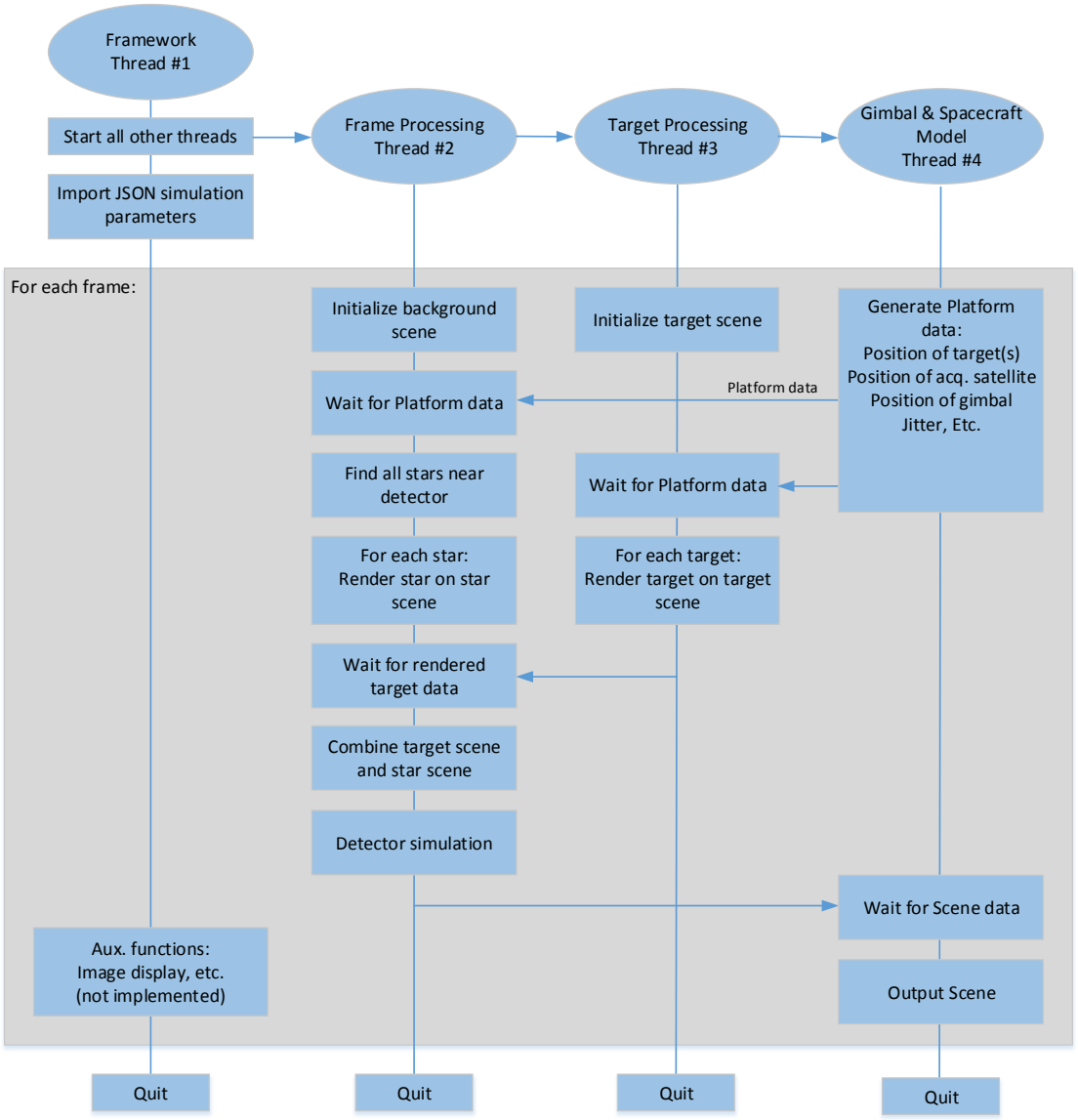

Exhibit 11: Top Level Software Processing Architecture.

Thread	Description
Framework	Main execution thread: -Ingests the simulation JSON Parameters -Starts all the remaining threads -Responsible for auxilary functions (displays/ image metadata / status)
Gimbal and Spacecraft	-Simulation of the imaging platform and targets -Interface between RT-PROXOR™ and HWIL input/output -Current frame simulation parameters are derived in this thread (such as time broken up into integration subsamples)

	-Pointing line of sight (LOS) vectors of the gimbaled sensor as well as positions of all objects as a function of the integration subsample are generated. -Low and high frequency jitter is computed as a function of the integration subsample and applied to the LOS.
Frame Processing	-Initial star catalog culling based on the LOS throughout the integration period. -Apply optical effects and LOS to each culled star and place on the focal plane. -When data is returned from the Target Processing: -Combine each target chip with the rendered star field -Apply detector modeling effects
Target Processing	-For each target, generate illumination angles and render a radiometrically correct lambert sphere. -Apply optical effects to each target chip. -Interpolate the target to the right size based on field of regard and distance.

Exhibit 12: Architecture Overview

The goal of the simulation architecture outlined is to be lightweight (low overhead threads with mutex based synchronization offloading the bulk of the processing to GPU resources), and expandable in the future as key modular components are improved/expanded upon. The above runtime architecture uses a modular object oriented C++ design to implement the design objectives. The simulation "world" is broken down into logically inherited components and subcomponents that are intended to be easily swapped in a plug and play manner when future improvements are envisioned (such as a different detector model, or a completely different approach of rendering stars).

ANALYSIS AND RESULTS

Verification under this development spiral has mainly been done at the unit test level. The key components (such as the detector model/optical model/target model/etc…) have mex interfaces to directly call the functional blocks from Matlab, and compare RT-PROXOR™ results to the Matlab equivalent model. The Matlab version of PROXOR™ has been extensively independently tested to verify radiometry, as well as validated against real satellite data for pointing and composite starfield rendering.

The current RT-PROXOR™ implementation is, for worst case conditions, at least a factor of 300 better than the original PROXOR™. Although realistic scenario cases are of more general interest, a worst case scenario was developed to compare the frame execution time associated with the various releases of the RT-PROXOR™ software package and assess overall timing improvements. This entails staring at a particularly dense region of the Milky Way galaxy and performing a high rate slew. Internally, the star rendering algorithm needs to keep track of and render each star at each integration subsample. A scenario like this saturates the number of GPU cores utilized in the frame processing, demonstrating worst case star rendering performance. The following figure (Exhibit 13) denotes a 2048 by 2048 focal plane, with a two second integration period, 501 subsamples per integration period, roughly 3220 stars apparent in the field of view, with a high angular rate slew causing severe streaking in each star.

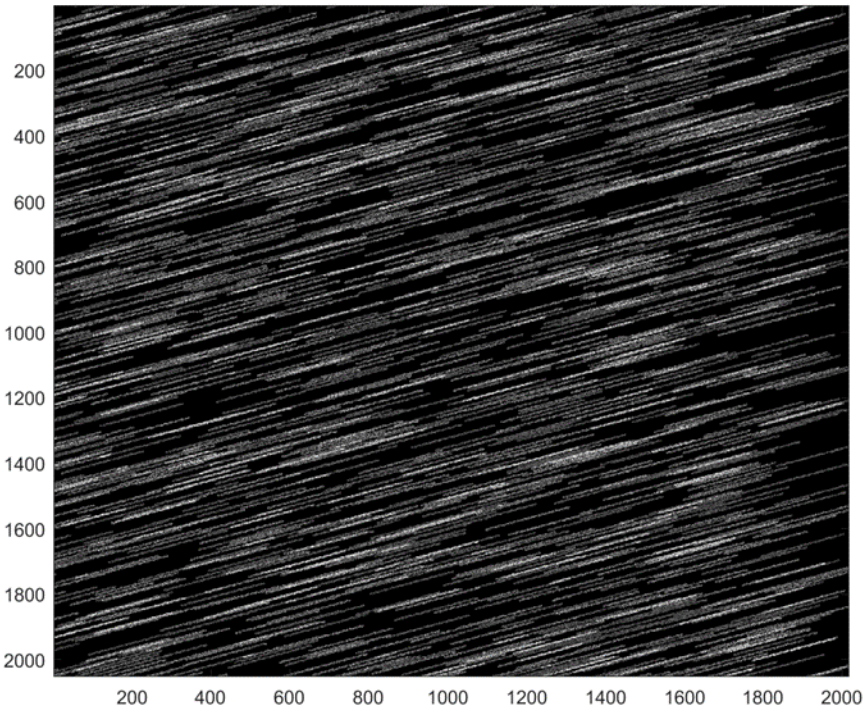

Exhibit 13: Simulated image with worst case streaking is used for benchmarking

The following table (Exhibit 14) highlights the frame execution time of this particular scenario through various releases of the RT-PROXOR™ software package:

Code Baseline	Code Description	Average Single Frame Execution Time (sec)	Effective Frame Rate (Hz)
Iter 1	Original Matlab PROXOR™ 2017 (not designed for speed)	305	0.003278689
Iter 2	Optimized Matlab RT-PROXOR™ 2017 IR&D	115	0.008695652
Iter 3	Hybrid Matlab RT-PROXOR™ with C++ Star Rendering, 2017 IR&D	18.2	0.054945055
Iter 4	GPU Based C++ RT-PROXOR™, 2018 IR&D	4.4	0.227272727
Iter 5	GPU Based C++ RT-PROXOR™, 2019 Optimized Multi-Kernel Star Rendering Algorithm	0.744	1.3441

Exhibit 14: Execution times for scenario with 2048x2048 FPA, 33x Spatial Oversample Factor, 501x Temporal Oversample Factor, and Multi-Kernel Star Rendering

The following figure (Exhibit 15) shows the performanceresults in graphical form.

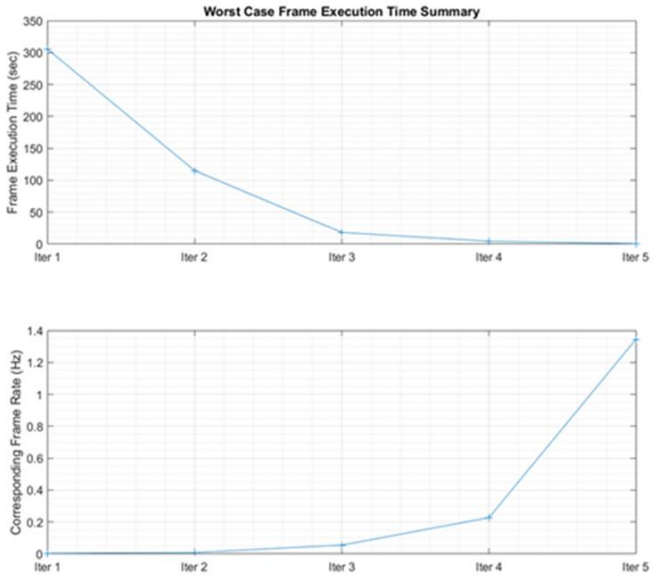

Exhibit 15: Worst Case Execution Time / Frame Rate Results

To further quantify performance, a subset of parameters were varied and benchmarked to show execution speed under a variety of realistic test conditions. There were four different simulations that were setup for the benchmarking tests, one 'typical' long range acquisition setup (Scenario 1), and three short range acquisition setups with varying slew rates (Scenarios 2A, 2B, 2C). The following figures (Exhibits 16 and 17) depict a rendered frame from each of these four configurations:

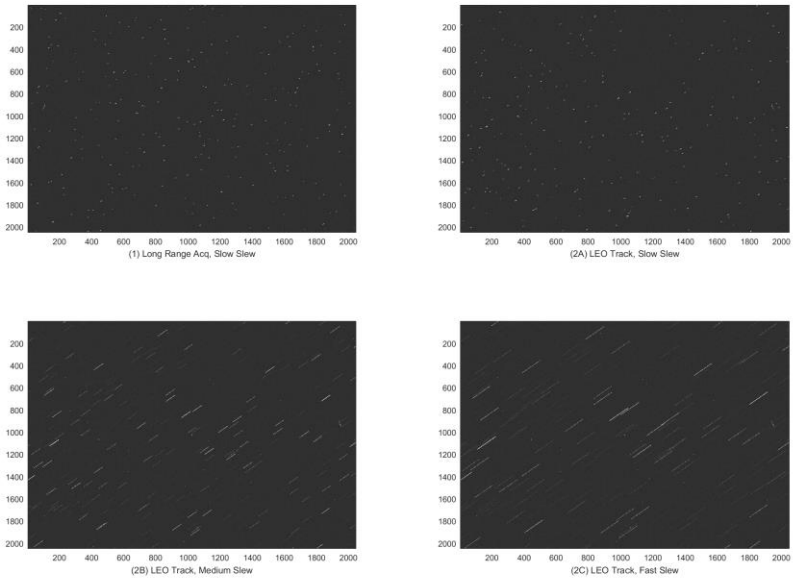

Exhibit 16: Four simulated images with varied realistic test conditions

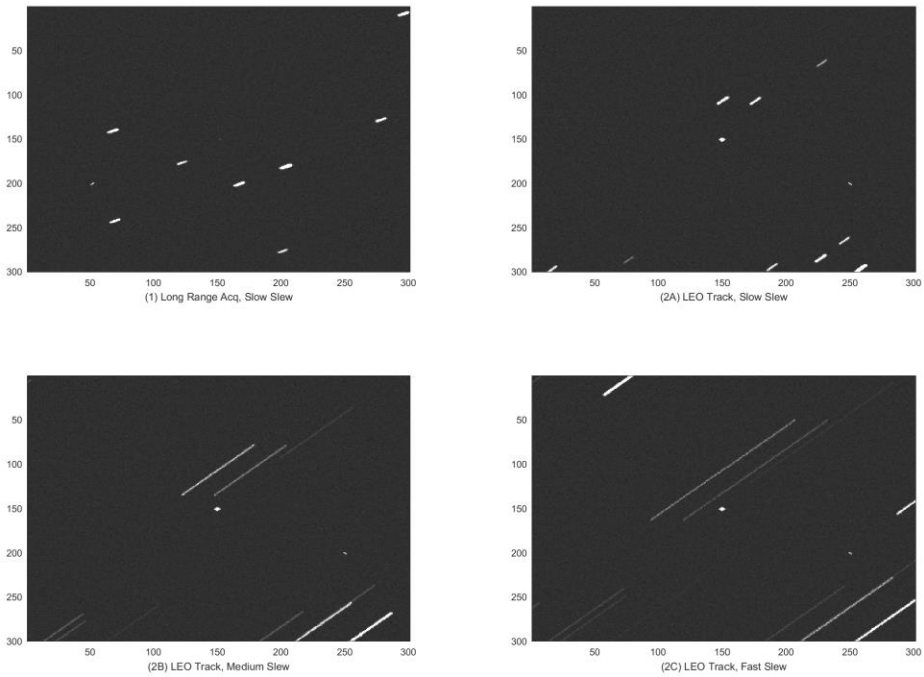

Exhibit 17: Zoom in of the four scenarios shows intra-frame streaking effects.**

The following figure (Exhibit 18) is a snapshot of the current "faster than realtime" scene generation performance speeds for a 2048x2048 array at a 2 second mission integration time at different temporal oversampling factors, as well as two different types of star rendering algorithms: single-kernel; and multi-kernel.

The two star rendering algorithms available determine how the LOS (line of sight) is calculated and applied to stars that are in the apparent field of view. In the single kernel case, a single LOS kernel is computed and applied to every star. This implies there is only linear motion of the acquisition sensor system, and all subsequent star tracks are therefore linear and the same shape. The multi-kernel approach calculates a unique LOS kernel for each star in the field of view. This case allows for both linear and rotational motion of the acquisition system, resulting in a different track for each star.

** Note: in scenarios 2A/2B/2C, the RSO of interest is being tracked and is visible in the middle of the frame.

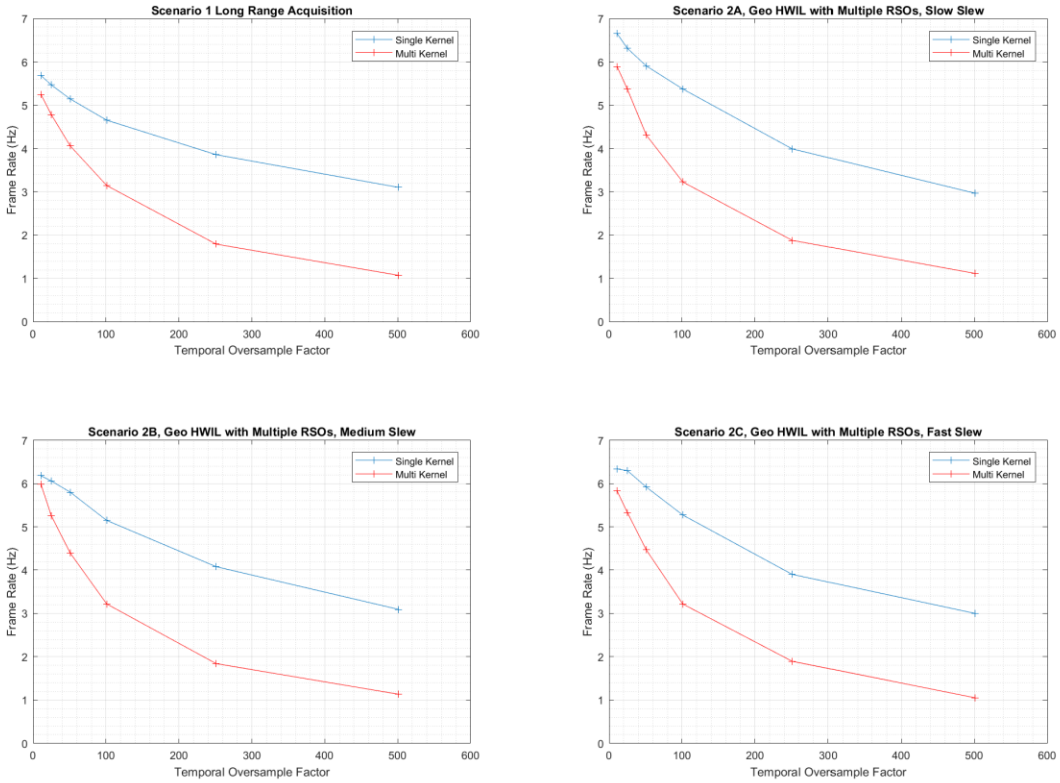

Exhibit 18: Faster than realtime performance speeds for a 2048x2048 array at a 2 second mission integration time

The oversampling factor dictates the intraframe fidelity of the simulation (subframes generated during the integration period to form the final composite image). For a scenario where there is little to no slewing, this parameter can be tuned to a small value with no loss in modelled performance. Future efforts will involve interactively tuning this parameter during a mission based on assessing motion of the start and end of integration quaternion to automatically adjust the subsampled fidelity. It would be very inefficient in terms of processing cycles to create 500+ temporal subsamples if there is minimal motion, and conversely too much intra-framing spatial smearing would be lost if a relatively high slew maneuver were initiated with a low number of subsamples.

Current benchmarking results are showing execution framerates up to 6-7 Hz. The benchmark results are very conservative, and continual improvements are underway to reach the desired objective of running at 30 Hz. Benchmarks were run on an NVIDIA-K6000 GPU, and going to a state-of-the-art Quadro P5000 GPU did improve the worst case scenario by 13%. As GPU technology (in terms of clock cycles and number of physical cores) continues to improve, the current design paradigm will scale the results accordingly to create better performance with the current code base as-is. Future efforts will also involve partitioning logical breaks in the processing to use its own dedicated GPU hardware. The target rendering module is already factored with this in mind (having its own independent processing thread in the current architecture).

Furthermore, each step of the processing chain is currently being benchmarked and assessed for performance and further optimization. Currently, the vast majority of the internal RT-PROXOR™ architecture is being run brute force on GPUs. However, GPU utilization is optimized for tasks that can be done in parallel through the usage of concurrent cores. Some portions of the processing chain are linear and not condusive to be optimized in such a manner, and may increase in speed if ran on the host target without external hardware. The flexibility of our architecture is condusive to testing blocks/modules on different targets, and NVIDIA has several assessment toolsets (such as the Visual Profiler) to provide itemized performance metrics.

The RT-PROXOR™ simulation world has over 100 user configurable parameters that affect the fidelity as well as runtime performance of the system. The oversample factor metric presented above is just one of the many factors that affect this. The results presented to this point entail "real-world" configured use cases developed at Ball for a variety of programmatic needs. Understanding the interaction between the various parameters may become especially important if achieving very high (60 FPS) frame rates proves difficult. It may be that the desired frame rate is attainable with a lower, but acceptable, fidelity system and thus additional herculean efforts may not be necessary to achieve the system requirements.

Alternatively, even if the highest fidelity system is always desired, understanding the interaction and the effect of each parameter on the fidelity may allow some of the parameters to be tuned back and still achieve the same, high fidelity system. Similarly, it may turn out that adjusting only a small number of these parameters up may increase the fidelity of the system with little computational cost. Being able to tune these parameters to adjust fidelity is part of our long-term approach and will be addressed in the future.

FUTURE RT-PROXOR™ DEVELOPMENT AND APPLICATION

Ball Aerospace's internal development efforts in 2018 took a giant step forward in re-imagining the PROXOR™ technology on a completely new platform and architecture. It was a major accomplishment rewriting the entire core code base in C++, and targeting key portions of the code to GPUs provided significant execution time improvements. The object-oriented redesign facilitates greater readability, organization, and expandability of the code base in the future. Our approach allows our code to autonomously improve as GPU technology improves in the future.

Based on the work done to date, there are several ongoing efforts to expand upon and improve the existing technology to facilitate future SSA mission planning/development as well as accommodate and assist existing programmatic objectives. The following outlines some of the ongoing efforts:

- Scenario Development - Having a SGP4 TLE based front end for satellite propagation allows us to model virtually any scenario imaginable with existing satellite databases. Additionally, having a hardware in the loop interface allows the capability to alter/perturb any scenario to simulate mission objectives. We are currently developing a user interface that will merge these two capabilities to allow a mission designer a flexible and visual front end to facilitate quick and effective scenario mockups.
- Extended Target Rendering - The full PROXOR™ simulation has the capability to use in-house as well as 3rd party renderers to do extended target imaging. Currently, RT- PROXOR™ is using Lambertian sphere modelling for point source rendering/target illumination, but future efforts will merge the ability to do full satellite rendering as well.
- Multi-GPU Development - Current development has been done on a single GPU resource (NVIDIA K6000, with 2048 cores), though there is no reason this cannot be expanded to use multiple GPUs simultaneously, or even a cloud based approach if needed (could conceivably be useful for extremely large focal plane sizes, where subtasks could not be completed in one pass with the number of cores available on hard resources). Currently, with the processing division, a single GPU resource has been identified as the best/most efficient path with respect to minimizing bottlenecks in the processing chain. As extended target rendering is integrated into the system, this will change, as it will require its own dedicated resources. Future efforts will assess the cost function of interprocessor GPU communication (passing data from one resource to another), vs executing additional processing kernels on a single GPU for optimized execution time of the processing chain. NVIDIA has proprietary card to card communication channels faster than PCI DMAs that will be explored with respect to this trade study.

RT-PROXOR™ is our tool of choice for existing and future space-based SSA and rendezvous missions. RT-PROXOR™ adds value with real-time high fidelity modeling of mission sensor data, enabling the government to solve challenging mission problems by allowing robust algorithm development and mission level analysis to meet their needs. Future development includes further model enhancements for increased speed and tailoring of the available fidelity level to work towards real-time simulation of these phenomena.

CONCLUSIONS

Ball has developed RT-PROXOR™ to address a critical need for supporting NRT/RT/Faster-than-realtime scene generation for Software In the Loop (SWIL) and Hardware In the Loop (HWIL) simulations in support of high-fidelity, extended duration, mission scenario testing of mission data processing algorithms. This testing results in robust algorithms that ensure mission success. It also increases the affordability of government programs by testing at high fidelity early in the program, where discovered issues can easily be rectified in a much shorter timeframe as compared to later in the program, where larger teams are involved and a higher level of process is used. This tool can be one piece in a broader mission level analysis tool that desires to include accurate detection and metric accuracy results based on actual algorithms running on high fidelity scenes. Typical desired frame rates of interest are about 4 – 30 frames per second (fps). This paper shows a snapshot of Ball's current efforts and details our path to achieving RT or faster-than-real-time performance. Our current performance for realistic scenarios with a 2048x2048 image format is 1-6 fps for multi-star kernels (each star has its own kernel) and 3-7 fps for fixed star kernels. For a non-realistic worst case scenario, used for benchmarking purposes, we showed a 300x improvement over our non-real-time version (PROXOR™). Employing a GPU-based architecture that automatically takes full advantage of the available GPUs, moving to a compiled language (C++) rather than a scripting language (Matlab), refactoring targeted areas of the code, and optimizing specific kernels has resulted in these significant speed improvements.

---

[1] Picciano, P, and Schurr, N, Enhanced Collaboration for Space Situational Awareness via Proxy Agents, *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference,* held in Wailea, Maui, Hawaii, September 11-14, 2012, Ed.: S. Ryan, The Maui Economic Development Board, id.85.

[2] Olivier, SS, A Simulation and Modeling Framework for Space Situational Awareness, LLNL-CONF-407013, Presented at: Advanced Maui Optical and Space Surveillance Technologies Conference, Wailea, HI, United States, Sep 16 - Sep 19, 2008.

[3] Hagerty, S, and Ellis, HB, An Innovative, High Fidelity Approach to Space Situational Awareness (SSA) Data Simulation, 33rd Space Symposium, Colorado Springs, April 2017

[4] Hagerty, S, and Ellis, HB, A High Fidelity Approach to Data Simulation for Space Situational Awareness Missions, *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference,* held in Wailea, Maui, Hawaii, September 20-23, 2016

