GPU-based Space Situational Awareness Simulation utilising parallelism for enhanced multi-sensor management

Tyler A. Hobson
I. Vaughan L. Clarkson
School of Information Technology & Electrical Engineering
The University of Queensland, Qld 4072, Australia

ABSTRACT

As a result of continual space activity since the 1950s, there are now a large number of man-made Resident Space Objects (RSOs) orbiting the Earth. Because of the large number of items and their relative speeds, the possibility of destructive collisions involving important space assets is now of significant concern to users and operators of space-borne technologies.

Consequently, a growing number of agencies are researching methods for improving techniques to maintain Space Situational Awareness (SSA). Computer simulation is a method commonly used to validate competing methodologies prior to full scale adoption. The use of supercomputing and/or reduced scale testing is often necessary to effectively simulate the intricacies of the SSA management problem.

Recently the authors described a simulation tool aimed at reducing the computational burden by selecting the minimum level of fidelity necessary for contrasting methodologies and by utilising multi-core CPU parallelism for increased computational efficiency. The resulting simulation runs on a single PC while maintaining the ability to effectively evaluate competing methodologies. Nonetheless, the ability to control the scale and expand upon the computational demands of the sensor management system is limited by the chosen architecture.

In this paper, we examine the advantages of increasing the parallelism of the simulation by means of General Purpose computing on Graphics Processing Units (GPGPU). As many sub-processes pertaining to SSA management are independent, we demonstrate how parallelisation via GPGPU has the potential to significantly enhance not only research into techniques for maintaining SSA, but also to enhance the level of sophistication of existing space surveillance sensors and sensor management systems. Even so, the use of GPGPU imposes certain limitations and adds to the implementation complexity, both of which require consideration to achieve an effective system. We discuss these challenges and how they can be overcome.

We further describe an application of the parallelised system where visibility prediction is used to enhance sensor management. This facilitates significant improvement in maximum catalogue error when RSOs become temporarily unobservable. The objective is to demonstrate the enhanced scalability and increased computational capability of the system.

1. INTRODUCTION

Since the late 1950s the number of man-made objects orbiting the Earth has been steadily increasing. As a result, the possibility of destructive collisions involving both manned and unmanned space missions is now of significant concern to users and operators of space-borne technologies. Whilst it is possible to predict collisions and attempt avoidance, success is reliant on how accurately the objects can be tracked.

In response to growing concern, agencies in the USA, China, Europe and Russia are conducting research to improve techniques for maintaining Space Situational Awareness (SSA) [1-4]. To reduce the cost of the research, agencies routinely employ computer simulation for assessing and comparing competing methodologies. Notably, the United

This work was supported by AFOSR grant FA9550-10-1-0493.

States Strategic Command (USSTRATCOM) and the European Space Agency (ESA) have established complete-loop simulated SSA systems to assist in this process [5, 6]. The large computational effort is supplied by supercomputing capable of exploiting the inherent parallelism of propagating the orbits of an extensive catalogue of Resident Space Objects (RSOs) and managing multiple sensors.

Recently the authors have proposed a MATLAB-based simulation on a single PC capable of a small degree of parallelism through its multi-core processor [7]. It was shown that a single PC using simplified physical models could successfully compare and contrast competing methodologies while permitting efficient, fast-paced research. However, such a system is not capable of simulation of RSOs on the scale that is undertaken daily in real-world space surveillance systems such as the United States' Space Surveillance Network. The simulated catalogue is necessarily smaller by some orders of magnitude.

In this paper, we extend this system to make it capable of a much larger degree of parallelism through the use of General Purpose computing on Graphics Processing Units (GPGPU). Recent developments in GPGPU hardware and associated Application Programming Interfaces (APIs) enable off-the-shelf consumer-grade systems to perform the parallel computing necessary for much more ambitious single-PC SSA simulations [8, 9].

Making use of GPGPU computing is, however, not without challenge. Mathematics libraries taken for granted in other languages are not yet routinely available for GPU processing. Parallel error trapping and the transportation of information to and from the GPU also require consideration. The use of highly parallel computation on a GPU requires a different approach to algorithm design, requiring a trade-off between computation and communication. We describe how the simulation system has been augmented with highly parallel, GPU-executed code to enable large-scale simulation.

We further describe an application of the parallelised system to demonstrate how parallelism permits the use of information often too computationally expensive to obtain. In this case, visibility forecasting is used to enhance the scheduling method by enabling the system to make more informed decisions about the value of observing elusive targets while they are visible. This example is intended to show that GPGPU is not only a valuable tool for SSA related research, but that it also has great potential for use in existing sensor management computers and intelligent space sensors by enabling them to make smarter and faster decisions.

Section 2 begins by providing an overview of GPGPU and what it offers in an SSA context. Subsequently, Section 3 describes the parallelised simulation system and highlights a number of areas in which the GPGPU architecture influenced its development and functionality. Section 4 provides a baseline understanding of how the simulation behaves without the enhancements proposed in Section 5, where details of the enhancements are discussed and results are produced. The concluding Section 6 discusses the outcome and the suitability of GPGPU for any number of applications aimed at improving SSA.

2. GPU COMPUTATION

GPGPU is a method of increasing the computational performance of a PC for scientific and engineering applications by utilising a Graphics Processing Unit (GPU) for large scale parallel processing applications. Although the concept has been around since the late 1970s, GPGPU has seen a slow adoption rate until around 2003 when CPU manufacturers began struggling to maintain the rate of increase that chip-speeds had experienced over the preceding decades [9]. In the short term, CPU manufacturers have circumvented this problem to some extent through the introduction of multicore processors which allow small scale computational parallelism. In 2006, graphic computing company NVIDIA released a versatile Application Programming Interface (API) and Software Development Kit (SDK) for the express purpose of enhancing accessibility to GPGPU, enabling even greater levels of PC-parallelism. Although alternative GPGPU architectures are available, MATLAB's compatibility with NVIDIA's well established Compute Unified Device Architecture (CUDA) [10] made it an obvious choice to augment the authors' previous work utilising multi-core parallelism for an SSA maintenance simulation in MATLAB [7].

MATLAB's CUDA-interface enables the transportation of data to and from a PC's conventional RAM and GPU-memory. In addition, it can initiate the execution of relatively low level C code on the GPU's many CUDA processing cores for fast parallel processing. The parallel-executed C code is processed on the GPU as one of many 'kernels'. Each kernel performs the same task as the next. However, each kernel has a unique identity enabling the programmer to address different parts of GPU-memory relative to the kernel ID. In theory this enables a single task, which would typically reside in one or more nested for-loops for CPU execution, to be applied to each of the values in a large array or matrix and executed once, in parallel. In reality the GPU has a finite number of parallel cores and therefore has a limit to the amount of parallel execution. Nonetheless, as described more completely in the CUDA programming guide [8], CUDA's Scalable Programming Model separates groups of kernels into 'blocks'. Each GPU has a finite number of blocks it can execute in parallel, on each CUDA-core. If there are more blocks than CUDA-cores, the GPU will sequentially process as many blocks as there are cores in parallel. Therefore, regardless of the size of the GPU, the task is executed in as parallel a manner as the hardware permits.

Importantly, although utilising even a modest GPU can achieve very high numbers of operations per second, the advantage over CPU computation is only gained if the problem is parallelisable on a large scale. This means GPGPU is not the right answer for many scientific and engineering tasks. Fortunately for SSA management, because of RSOs' comparably minor masses to Earth and their typically large separation, the fundamental problem can be reduced to many individual sensors observing a large population of independently orbiting RSOs. This enables large scale parallelisation and suitability for GPGPU. If however the increased process error due to independence assumptions is deemed unacceptable, work by Nyland et al. [11] demonstrates the GPU's ability to also enhance n-body simulations. In addition to independence of RSO motion and sensor tasking, under certain conditions the solution can also be parallelised in time if closed-form orbit propagators are suitable. Large-scale parallelisation of tasks such as visibility analysis, observation effectiveness determination, orbit propagation, catalogue error analysis, observation simulation and orbit determination are therefore tractable, making SSA management and simulation an excellent match for GPGPU. For these reasons, all of the aforementioned SSA applications have been implemented as GPU kernels and used to produce results for this paper.

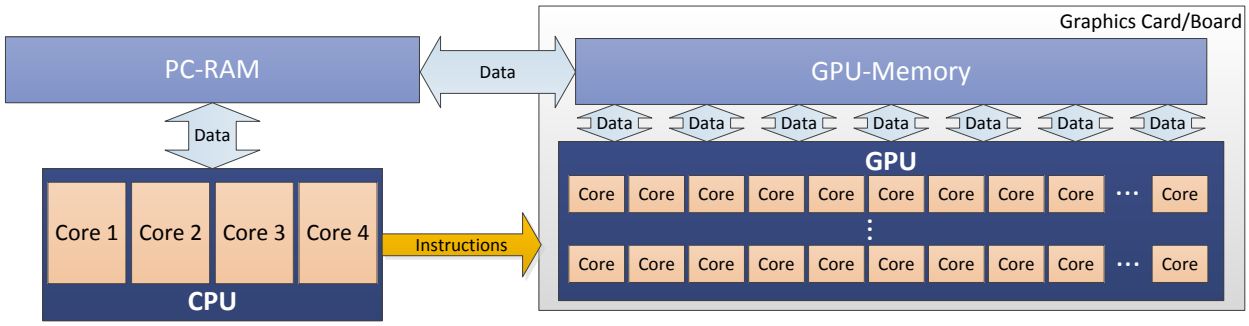

Fig. 1. - High-level GPGPU system architecture

Although GPGPU holds great promise for SSA management, there are a number of limitations and hurdles that must be judiciously managed to achieve an effective system. GPGPU applications require large scale independence primarily due to the time it takes to transfer information between PC-RAM and GPU-memory. Although it may take a long time for a CPU to process a large amount of information, the information is typically already stored in PC-RAM enabling computation to begin immediately. Conversely, GPGPU usually requires data to be transferred to and from GPU-memory before and after processing. This means the time saved performing parallel operations must be sufficiently great to warrant the time taken for transferring the data. Tasks involving large amounts of data but only a small amount of parallelism and/or processing are therefore best left to one or more CPUs. Nonetheless this limitation can be alleviated in some cases by recomputing known values on the GPU and leaving arrays and matrices of data stored on the GPU to minimise transference of information. In the case of SSA simulation, leaving

sensor and RSO state information in GPU-memory has proved to be advantageous. This methodology however presumes the GPU has sufficient memory to store the information throughout the simulation.

Additional limitations to GPGPU are a consequence of the late-blooming of the technology. Although basic math libraries have been rewritten in device compatible C code, higher level libraries taken for granted in well-established programming languages are yet to be transitioned into GPU-device executable code for single thread execution. For this reason a tailored matrix-mathematics library ranging from basic matrix operations to more sophisticated matrix inversion and Cholesky decomposition had to be developed for this simulation. Furthermore, to permit the transfer of debugging and error information from the GPU, additional and often redundant data had to be transported with each parallel execution due to a lack of structured error handling. CUDA code is also hardware specific and needs to be compiled with a minimum architecture version in mind in order to take advantage of the benefits of the version's attributes. For example, double precision processing, which has been crucial for producing the results of this paper, has only been possible in the later versions of CUDA and its compatible hardware. In addition, the double precision arithmetic does not fully comply with IEEE standards [12], leading to slight differences between CPU and GPU results. Nevertheless, it is anticipated that as the technology matures its imperfections will diminish in prominence.

3. SIMULATION METHODOLOGY

The simulation produced for this paper simulates a closed-loop space surveillance network. The simulation computes positions of orbiting and celestial objects to supply the fundamental geometric truth-data. A sensor management system then uses sensor models to observe and track a predefined catalogue of RSOs which contains estimated orbital state information. Sensors are scheduled by the sensor management system on a daily basis and it is assumed all sensor variants take 120 seconds to make five measurements which constitute a single observation. The observation data is used to update the RSO catalogue's state estimates which are then compared with the truth-data to determine the degree of SSA obtained.

3.1. The Simulated RSO Catalogue

Simulated objects were obtained from genuine Two Line Element (TLE) data publicly available on the Space-Track.org website [13]. The file 'all_sat_191.zip' containing all the TLE updates recorded on the 191st day of the year 2012 was downloaded and processed to extract the approximate orbital elements of each of the recorded objects. Although using this file alone does not guarantee a complete catalogue, 14876 unique objects were obtained. Fig. 2. shows the distribution of objects with respect to the semi-major axis length of orbit. Three distinct groupings are observable. The first grouping, comprising 80% of the catalogue, are the objects orbiting in Low Earth Orbit (LEO) between the Earth's atmosphere and approximately 2000km of altitude. The second grouping can be found at approximately the centre of the Medium Earth Orbit (MEO) region with an orbital period of approximately half a day. The final group can be found at the border of MEO and High Earth Orbit (HEO) at around 35786km of altitude with an orbital period of approximately a day.

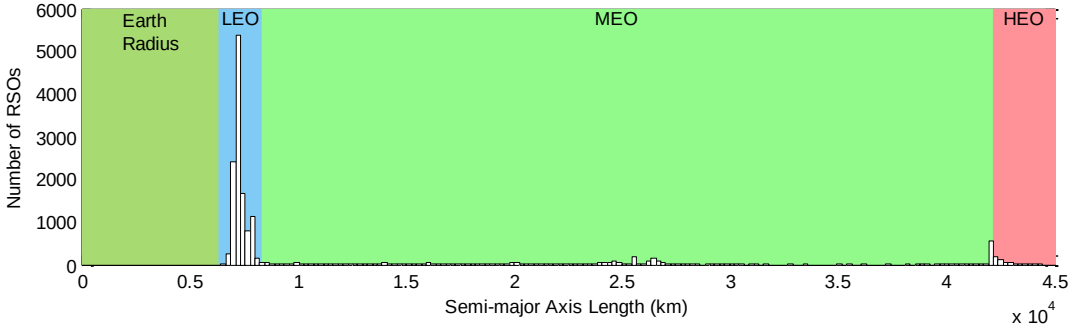

Fig. 2. Frequency of all catalogued RSOs verses semi-major axis length

The TLEs are compiled into an RSO catalogue by firstly converting all objects' orbital elements to a rectangular Earth-centred coordinate system, resulting in a six element state containing position and velocity components. The i^{th} target in the RSO catalogue at time $k = 0$ therefore has the truth-state vector

$$\underline{x}^i_{k=0} = [x \quad y \quad z \quad \dot{x} \quad \dot{y} \quad \dot{z}]^T. \tag{1}$$

The variances of a diagonal covariance matrix $\boldsymbol{P}$ are empirically devised to initialise the catalogue in a reasonably poor state of accuracy. The matrix is thereafter applied to and recorded for each truth-state to provide an initial state estimate $\hat{\underline{x}}^i_{k=0}$ and covariance $\boldsymbol{P}_{k=0}$ for all RSOs, where

$$\boldsymbol{P}_{k=0} = diag\left[\sigma^2_{pos}\ \sigma^2_{pos}\ \sigma^2_{pos}\ \sigma^2_{vel}\ \sigma^2_{vel}\ \sigma^2_{vel}\right], \tag{2}$$

$\boldsymbol{L}\boldsymbol{L}^{\boldsymbol{T}} = \boldsymbol{P}_{k=0}$ such that $\boldsymbol{L}$ is the Cholesky factorization of $\boldsymbol{P}_{k=0}$ and letting $\underline{\mathbb{r}}$ be a vector of six i.i.d. $\mathrm{N}(0,1)$ random numbers,

$$\hat{\underline{x}}^i_{k=0} = \underline{x}^i_{k=0} + \boldsymbol{L}\,\underline{\mathbb{r}}. \tag{3}$$

To monitor and compare the condition of the catalogue throughout a simulation, two metrics are obtained. The metrics are computed by firstly propagating all RSOs' state estimates to the end of the current day. All RSOs' estimated and truth orbits are then computed in parallel on the GPU for the subsequent 24 hours and the maximum position error is recorded. From the list of maximum errors, the catalogue maximum and median are obtained.

3.2. Orbit Determination

Once observations have been scheduled and simulated, state estimates are augmented with the new observational data by producing a best-fit estimated state via an appropriate non-linear recursive filtering process. Due to its previously demonstrated robust behaviour in the authors' recent work [7], the Unscented Kalman Filter (UKF) [14] was again adopted for this role. Use of the UKF has not only proven advantageous due to the UKF's ability to cope with the nonlinearity of orbital propagation, but also due to the relatively low computational complexity required when compared to more sophisticated filters such as a particle or Sequential Monte Carlo (SMC) filter [14].

A non-trivial observation queuing process enables simulated data to be applied in a catalogue-wide parallel update. Current state, covariance and update-epoch information remains on the GPU to enable all RSOs to be updated efficiently and simultaneously. In the event of simultaneous observations from multiple sensors, the system reverse-time propagates the RSO's state information by 120 seconds, and then applies the new observation as usual.

3.3. Sensor Scheduling

The scheduling method employed adopts a centralised sensor-management network-topology. The sensor manager holds the RSO catalogue, controls the global sensor network and collates sensor observations to update the catalogue. The sensor manager controls the sensors by creating a schedule for each sensor, telling them when to observe select RSOs during an upcoming scheduling period. For the creation of all results in this paper, the scheduling period has been arbitrarily chosen to be 24hrs. To create the schedule, a method for evaluating the observation effectiveness of an RSO/sensor/time combination throughout a scheduling period is used that exploits the RSO orbit error covariance information stored within the catalogue. Observation effectiveness is measured using the Kalman filter update equation which estimates the amount of reduction in orbit error covariance given a proposed RSO/sensor/time observation.

Observations $\boldsymbol{Z}$ of the i^{th} target are made by the j^{th} sensor such that

$$\underline{z}^{i,j} = \boldsymbol{H}^j \underline{x}^i + \underline{n}^j \tag{4}$$

where $\boldsymbol{H}$ is the observation model and $\underline{n}^j$ is the noise vector of the j^{th} sensor. Each noise component is assumed zero mean and each sensor's noise auto-covariance matrix is defined

$$\boldsymbol{R}^j = E\left[\underline{n}^j \underline{n}^{j^T}\right]. \tag{5}$$

The Kalman Filter update equation of interest is the a posteriori estimate covariance

$$\boldsymbol{P}^{i}_{k|k} = \boldsymbol{P}^{i}_{k|k-\Delta k} - \boldsymbol{K}^{i,j}_{k}\boldsymbol{H}^{j}_{k}\boldsymbol{P}^{i}_{k|k-\Delta k} \tag{6}$$

where

$$\boldsymbol{K}^{i,j}_{k} = \boldsymbol{P}^{i}_{k|k-\Delta k}\boldsymbol{H}^{j^{T}}_{k}(\boldsymbol{H}^{j}_{k}\boldsymbol{P}^{i}_{k|k-\Delta k}\boldsymbol{H}^{j^{T}}_{k} + \boldsymbol{R}^{j}_{k})^{-1}\,. \tag{7}$$

The time k represents the epoch of observation while Δk represents the time since the last state estimate was made. Of significance is the final term in Eq. (6). The matrix $\boldsymbol{K}^{i,j}_{k}\boldsymbol{H}^{j}_{k}\boldsymbol{P}^{i}_{k|k-\Delta k}$ represents the predicted reduction in the a priori covariance of the i^{th} RSO due to a measurement update, made by the j^{th} sensor. Although the question of optimality is open, the method employed by Hill et al. [5] is followed. The observation-effectiveness metric $\boldsymbol{\beta}_{red}$, is subsequently computed by taking the trace of the upper left position quadrant such that

$$\boldsymbol{\beta}^{i,j,k}_{red} = tr\left(\left[\boldsymbol{K}^{i,j}_{k}\boldsymbol{H}^{j}_{k}\boldsymbol{P}^{i}_{k|k-\Delta k}\right]_{p,3\times 3}\right). \tag{8}$$

The objective is to assess $\boldsymbol{\beta}_{red}$ for all RSO/sensor/time combinations so that all possibilities may be compared. However, a combination's $\boldsymbol{\beta}_{red}$ value is only relevant if the sensor has the ability to observe the RSO at the proposed time. Therefore in practice, $\boldsymbol{\beta}_{red}$ is assessed in conjunction with sensor models producing a three-dimensional solution space exhibiting regions of sparsity due to a lack of visibility. Thereafter, the most effective observation can be scheduled by locating the maximum value. Once the first preference is selected, Eq. (6) is evaluated to obtain the estimated a posteriori covariance, followed by its reverse time propagation back to the beginning of the scheduling period. This permits the re-evaluation of $\boldsymbol{\beta}_{red}$ for the chosen RSO at all sites and times, while compensating for the newly scheduled event. The process is thereafter repeated until the schedule for each sensor is full.

The evaluation of RSO-sensor visibility and $\boldsymbol{\beta}_{red}$ are both examples of independent processes. RSO visibility was therefore pre-computed on the GPU for the entire scheduling period. Subsequently, the visibility data is used in conjunction with the GPU to evaluate the initial $\boldsymbol{\beta}_{red}$ solution space and re-evaluate $\boldsymbol{\beta}_{red}$ when an observation is scheduled. However, because re-evaluation of $\boldsymbol{\beta}_{red}$ modifies the solution space and it is assumed sensors can only observe one RSO at a time, scheduling each observation is dependent on the previous event. The scheduling process is therefore necessarily sequential and is by far the slowest process of all due to the lack of parallelism. While parallelised processes take seconds to complete, scheduling currently requires minutes of processing per thousand observations. This trait means the simulation runtime is very weakly influenced by the size of the catalogue and is instead heavily dependent on the number of sensors and the cumulative sum of the number of observations permitted per scheduling period.

3.4. Sensor Modelling

Interchangeable sensor models have been developed to produce the types of measurements commonly made by space sensors. Sensors can be configured to measure all or any subset of the parameters range, azimuth, elevation and their rates, relative to the sensor and the observed RSO. Sensors can also assume an active or passive sensing regime, which alters the requirements for determining visibility of an RSO. In general, it is assumed a minimum elevation of 20°, line-of-sight and a maximum range is all that is necessary for determining visibility by an active sensor such as radar or for lasing. In addition to these requirements, passive sensing requires knowledge of the sun-RSO-sensor geometry, local atmospheric lighting conditions and Earth shadowing as it is assumed the sun is the source of the reflected radiation.

4. SENSOR CHARACTERISATION

To evaluate and understand the behaviour of the simulation when alternative sensors are used, a number of homogeneous sensor network simulations are run. Using this method, any weaknesses of each sensor type are exaggerated and decoupled from the attributes of the other types. Homogeneous optical, radar and Doppler radar

sensor networks are compared. Measurement characteristics of each sensor type are summarised in Table 1. All error parameters are assumed i.i.d. Gaussian and are stated in standard deviation.

Table 1. Sensor Measurement Characteristics

Sensor Type	Range Error	Angular Error	Range-Rate Error	Maximum Range
Optical	-	1 arc second	-	∞
Radar	30 m	50 arc seconds	-	8300km
Doppler Radar	30 m	50 arc seconds	*variable*	8300km

These values were chosen as representative mid to high-end values of actual sensors [15, 16]. The range rate of the Doppler radar is varied to examine the effect of adding extra information to the radar measurement.

The results of a number of 8-day simulations, using 8 sensors arbitrarily located about the globe, of all types and using a series of range-rate values for the Doppler radar, are displayed in Fig. 3. Catalogue maximum error values are consistently high as in all cases there are a large number of objects that the homogeneous sensor network does not observe. Radar fails to observe MEO/HEO objects due to its range limitations while optical fails to observe LEOs due to solar eclipsing and reliance on nightfall at the sensor. This effect is again evident when comparing the Catalogue Maximum Error between optical and radar types.

As suggested in the authors' previous work [7], when radars are limited to making the same number of observations as optical sensors can passively achieve, the optical sensor's high precision actually outperforms radar with respect to Catalogue Median Error. Nevertheless, Fig. 3. also shows that when radar is unrestricted it is capable of achieving a much higher rate of observations per day due to active sensing and makes up for its reduced accuracy. Furthermore, the addition of range-rate data to a radar's measurement shows that with sufficiently precise data, catalogue accuracy can be further improved. This data suggests space surveillance networks should be comprised of sophisticated multi-measurement radars while keeping a small number of optical sensors for observing near HEO RSOs.

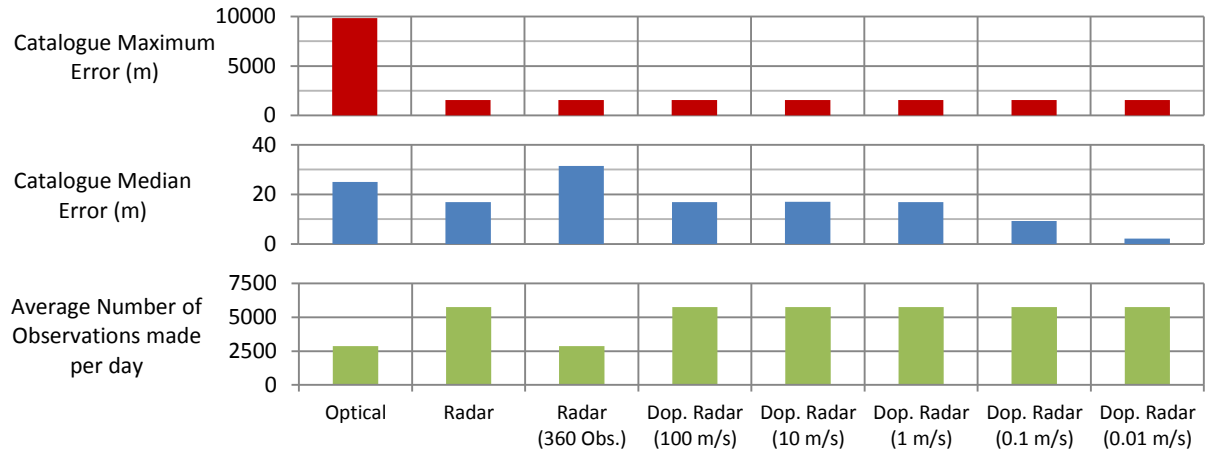

Fig. 3. 8-day, 8-Sensor homogeneous-network simulations

Conversely, as a result of diminishing costs and increases in accuracy of electro-optical sensors, the literature suggests increasing the use of optical space sensors [5, 16-18]. It is proposed that in addition to the accuracy and versatility afforded by a small and potentially mobile electro-optical sensor, they are also much cheaper to acquire and operate than the advanced Space Surveillance Radars presently in use. It is therefore advantageous if new scheduling techniques could be developed to enhance the effectiveness of a space surveillance network with a high optical to radar ratio. While it may be possible to overcome the reduced number of observations per day by widely dispersing many cheap optical sensors, the reduced effectiveness due to reduced routine LEO observations requires consideration.

5. VISIBILITY PREDICTION APPLICATION

5.1. Enhanced Observation Effectiveness Scheduling via Visibility Prediction

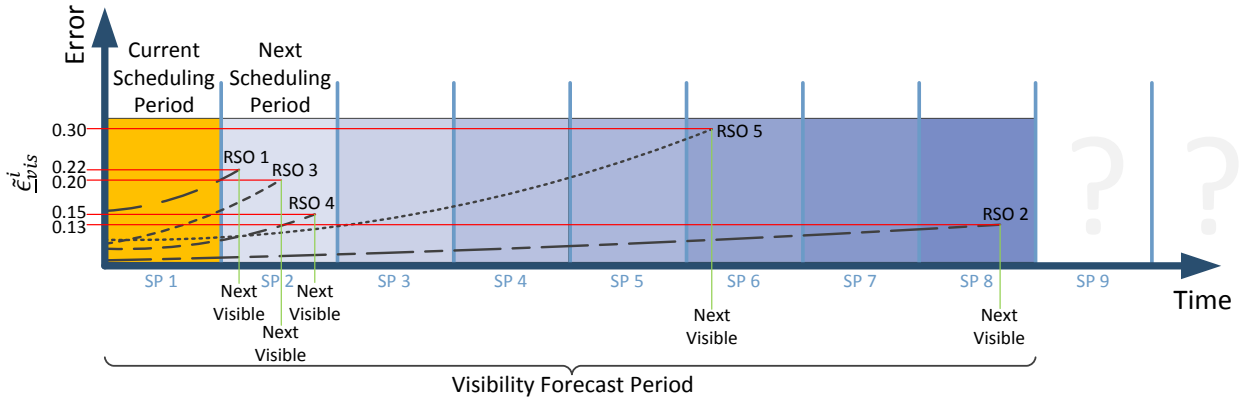

Fig. 4. Observation effectiveness scaling via visibility prediction

In spite of how effective covariance based scheduling has been shown to be [5, 7], it is also acknowledged that the method is a greedy, sub-optimal solution. The $\boldsymbol{\beta}_{red}$ metric does not anticipate long periods of RSO invisibility. As depicted in Fig. 4, after an RSO is observed, any observation error or process error will cause the variance of its state estimate to grow with time. If an RSO is not visible for an extended period of time, its variance can grow so dramatically that its estimated orbital parameters become unreliable for the purposes of conjunction analysis and cannot be reacquired by a sensor. Therefore, we propose to compensate for these periods by prioritising observation of the RSO just prior to the period of invisibility.

Algorithm 1. Enhanced observation effectiveness scaling

Parallelisable Variables	Pseudo Code	Comment
$\boldsymbol{i}, \boldsymbol{j}$ & $\boldsymbol{k}$	$\boldsymbol{\nu}_{bin}^{i,j,k} = \text{CHECK\_VISIBILITY (i, j, k)}$	Determine when each object is next visible by the sensor network during the visibility forecast period: $\boldsymbol{\nu}_{bin}$ – is a three-dimensional binary matrix indicating visibility. $\boldsymbol{\nu}_{fs}$ – is a two-dimensional matrix recording the time t_k of the first instance of RSO-Sensor visibility. $\underline{\nu}_{next}$ – is a vector storing each RSO's earliest time of visibility for the entire network. MIN_j – returns the earliest instance of visibility amongst all of the sensors.

$\boldsymbol{i}$ & $\boldsymbol{j}$	$k \equiv +1day$
	LOOP
	IF $(\boldsymbol{\nu}_{bin}^{i,j,k} = TRUE)$ OR $(k \equiv +8days)$
	EXIT LOOP
	ELSE
	$k = k + 1$
	END IF
	END LOOP
	$\boldsymbol{\nu}_{fs}^{i,j} = t_k$	

$\boldsymbol{i}$	$\underline{\nu}_{next}^{i} = \text{MIN}_j(\boldsymbol{\nu}_{fs}^{i,j})$	
$\boldsymbol{i}$	$\boldsymbol{P}_{k-\Delta k\|k-\Delta k}^{i} \rightarrow \boldsymbol{P}_{k\|k-\Delta k}^{i}$; where $\Delta k = \underline{\nu}_{next}^{i}$ $\underline{\epsilon}_{vis}^{i} = tr\left(\left[\boldsymbol{P}_{k\|k=0}^{i}\right]_{p,3\times3}\right)$	Propagate $\boldsymbol{P}^{\boldsymbol{i}}$ from the start of the scheduling period to $\underline{\nu}_{next}^{i}$ using standard Kalman filtering propagation and record $\underline{\epsilon}_{vis}$.
None.	$\tilde{\underline{\epsilon}}_{vis}^{i} = \frac{\underline{\epsilon}_{vis}^{i}}{\sum_{i=1}^{N_{RSO}} \underline{\epsilon}_{vis}^{i}}$	Normalise $\underline{\epsilon}_{vis}$, where N_{RSO} is the total number of RSOs.
$\boldsymbol{i}$	$\tilde{\boldsymbol{\beta}}_{red}^{i} = \boldsymbol{\beta}_{red}^{i} \cdot \tilde{\underline{\epsilon}}_{vis}^{i}$	Apply scaling to the i^{th} layer of $\boldsymbol{\beta}_{red}$.

Due to the increased computational power provided by the GPU, it is proposed that the system compute visibility for every object relative to every sensor well into the future, so the system can anticipate periods of RSO-invisibility and compensate accordingly. As summarised in Algorithm 1, at the beginning of a scheduling period the time until an object is next visible to the sensor network, after the current scheduling period, is determined. The orbit error

covariance for each object is then propagated forward to this point via a Kalman time update and recorded. Similar to the $\boldsymbol{\beta}_{red}$ metric, the trace of the position component is used to obtain the ϵ_{vis} metric for comparison. Finally, all ϵ_{vis} values are normalised and used to scale $\boldsymbol{\beta}_{red}$ for the current scheduling period. This in turn causes the $\boldsymbol{\beta}_{red}$ value of an object that is about to become invisible to the network to be scaled according to its estimated variance when it is next visible to the network. In practicality visibility cannot be computed indefinitely so an 8-day visibility forecast period was empirically chosen as a manageable length of time. If an object is not visible in this time, it is assumed to be visible at the end of this period.

5.2. Method Evaluation via Simulation

As it is expected that a sensor network with a high ratio of optical to radar sensors will experience periods of reduced visibility of LEO objects, it is hypothesised that the enhancements described will improve the suitability of the covariance based observation effectiveness metric for such a network. To test the hypothesis, the simulation was configured with 17 optical sensors and 1 radar to ensure the time between sightings was unlikely to be longer than the 8-day visibility forecast period. Sensors were positioned all about the globe in locations such as America, Australia, Europe, South Africa, South America and the Pacific Ocean. Two 24-day simulated scheduling scenarios were run for comparison. Scenario A used the predefined $\boldsymbol{\beta}_{red}$ observation effectiveness method to schedule observations for its sensors. Scenario B utilised visibility prediction to enhance the observation effectiveness method.

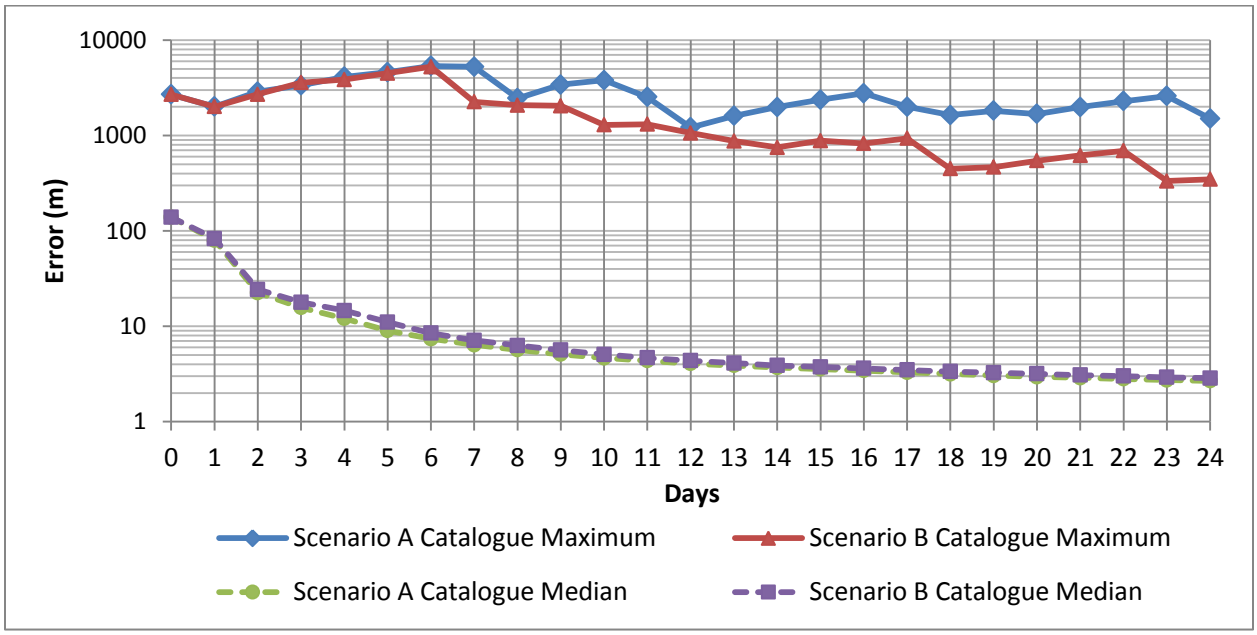

Fig. 5. Scenario comparison of a 24-Day simulation with and without enhanced scheduling

Fig. 5. displays the results of these simulations on the same logarithmic error plot. The most obvious influence of Scenario B can be seen upon comparison of each scenario's Catalogue Maximum Error. After approximately 6 days, the Scenario B maximum begins to diverge from Scenario A's and continues diverging until the end of the simulation when Scenario B's maximum is an order of magnitude more accurate than Scenario A's.

A small separation is also observable between each scenario's Catalogue Median Errors. Scenario A's median appears slightly more accurate than Scenario B's however the discrepancy is minor and the shape of the curve, describing the catalogue's improving accuracy, is shared.

6. CONCLUSION

The results suggest that the use of the enhanced-scheduling method may degrade the Catalogue Median Error as a result of an increased focus on any elusive RSOs. Nonetheless the reduction in accuracy is very minor while the consequential improvement in maximum catalogue error is significant. In addition, although network visibility prediction has been presented as a benefit to passive-sensing networks, the same method could be used to improve active-sensing networks. Predictive information such as routine maintenance, inclement weather predictions or any other foreseeable outage event could be used to prepare the catalogue for periods of reduced visibility.

When utilising the enhanced observation effectiveness scheduling method, the computationally intensive predictive visibility analysis is performed prior to the sequential scheduling. Because the visibility analysis is performed in parallel, the process is very fast and will very weakly affect the run time of the simulation regardless of the number of catalogued RSOs. The comparison of the test scenarios therefore demonstrates how valuable the increased computational power of the sensor manager can be, while only lightly influencing the run time and maintaining scalability due to parallelism.

Due to the success of this preliminary application of GPGPU for managing SSA and due to the scale of the simulation itself, it is hoped that this work encourages further research into methods of leveraging the benefits of parallelised algorithms. In spite of the growing number of RSOs orbiting the Earth, emerging technologies such as GPGPU could enable the production of smarter and faster sensor management systems to deliver a commensurate growth in tracking capability.

7. REFERENCES

1. Ender, J. et al, *Radar techniques for space situational awareness*, 2011 Proceedings International Radar Symposium (IRS), 7-9 Sep, 2011
2. Erwin, R.S. et al, *Dynamic sensor tasking for Space Situational Awareness*, 2010 American Control Conference (ACC), Baltimore, Maryland, USA, Jun 30th - Jul 2nd, 2010
3. Fateev, V.F. et al, *Analysis of Collision Prediction Characteristics*, Eighth US/Russian Space Surveillance Workshop Space Surveillance Detecting and Tracking Innovation, 2010
4. Mingyan, C. et al, *Study on simulation of signal processing system of space surveillance radar*, 2008 International Conference on Radar, 2-5 Sep, 2008
5. Hill, K. et al, *Covariance-Based Network Tasking of Optical Sensors*, in Proc. AAS/AIAA Space Flight Machanics Meeting, Feb, 2010
6. Sánchez-Ortiz, N. et al, *AS4, a simulator supporting the definition of the European space surveillance segment of SSA*, European Space Surveillance Conference, INTA HQ, Madrid, Spain, 7-9 Jun, 2011
7. Hobson, T.A. and Clarkson, I.V.L., *Sensor-scheduling simulation of disparate sensors for Space Situational Awareness*, Advanced Maui Optical and Space Surveillance Technologies Conference, Maui, HI, 13-16 Sep., 2011
8. NVIDIA Corporation *NVIDIA CUDA: Programming Guide*. **4.0**. [cited 31st Oct 2011]; Available from: http://developer.nvidia.com/nvidia-gpu-computing-documentation, 2011
9. GPGPU .org. *General-Purpose Computation on Graphics Hardware*. [cited 2011 25th Oct]; Available from: http://gpgpu.org/. 2012
10. The MathWorks, I. *MATLAB GPU Computing Support for NVIDIA CUDA-Enabled GPUs*. [cited 2011 20th Oct]; Available from: http://www.mathworks.com.au/discovery/matlab-gpu.html. 2012
11. Nyland, L. et al, *Fast N-Body Simulation with CUDA.* GPU Gems 3, 2007(Second printing): p. Chapter 31.Dec. 2007
12. NVIDIA Corporation *CUDA C Best Practices Guide*. **4.0**. [cited 31st Oct 2011]; Available from: http://developer.nvidia.com/nvidia-gpu-computing-documentation, 2011
13. USSTRATCOM. *Space Track*. [cited 2011 Oct 10th]; Available from: www.space-track.org. 2004
14. Ristic, B. et al, *Beyond Kalman Filtering*, Artech House, 2004
15. Vallado, D.A. and McClain, W.D., *Fundamentals of astrodynamics and applications*. 3rd ed. Space technology library, Springer, New York, 2007
16. Sabol, C., *A Role for Improved Angular Observations in Geosynchronous Orbit Determination*, PhD Thesis, University of Colorado at Boulder, 1998
17. Sabol, C. et al, *A Fresh Look at Angles-Only Orbit Determination*, AAS/AIAA Astrodynamics Specialist Conference, Girdwood, Alaska, 16-19 Aug, 1999
18. Thrall, M.L., *Orbit Determination of Highly Eccentric Orbits using a RAVEN Telescope*, Thesis, Naval Postgrauate School, Monterey, 2005