ABSTRACT

A new radiometry and design framework has been introduced in the latest Digital Imaging and Remote Sensing Image Generation model (DIRSIG5) that allows for faster simulations while streamlining the generation of high-fidelity radiometric data. The same framework that allows for improved computational performance has also modularized simulation components to allow for extensive interchangeability based on simulation needs. This new framework includes several plugin interfaces that facilitate native and 3rd party extensions to the model. A sensor plugin and its interaction with the internal radiometry engine is described and then demonstrated by modeling systems with rolling shutters and time-delayed integration.

Index Terms— DIRSIG, Data modeling, Modeling and simulation, Scene simulation, Sensors, Simulations, Software

1. MODEL INTRODUCTION

The initial development of the Digital Imaging and Remote Sensing Image Generation (DIRSIG) model began at Rochester Institute of Technology (RIT) in the late 1980s as a 3-D simulation environment for predicting images that would be produced by thermal infrared systems. Since that time, the model has been expanded to cover the 0.2 to 20.0 micron region of the spectrum. The model is designed to generate passive broad-band, multi-spectral, hyper-spectral, low-light, polarized, active laser radar datasets through the integration of a suite of physics-based radiation propagation modules. These modules address tasks ranging from bidirectional reflectance distribution function (BRDF) predictions of a surface, to time and material dependent surface temperature predictions, to the dynamic viewing geometry of scanning imaging instruments on agile ground, airborne and space-based platforms. The model also contains a suite of interfaces that leverage externally developed components (e.g. temperature prediction, atmospheric propagation, etc.) that are modeling workhorses for the multi- and hyper-spectral remote sensing community. The software is employed internally at RIT and externally by a user community as a tool to aid in the evaluation of sensor designs and to produce imagery for algorithm testing purposes. Key components of the model and some aspects of the model’s overall performance have been gauged by several validation efforts over the model’s evolution[1][2].

The last major update of the model, DIRSIG4, introduced a modern and flexible software architecture to support new sensor modalities and more complex scenes. Since that time, the needs of the user community have grown and diversified in tandem with advancements in the computational capabilities of modern hardware. Faced with a desire to model more complex, multi-component systems that are beyond the original intent and capabilities of an aging software design, a new version of DIRSIG, version 5, is being introduced to the community.

The overarching goal of DIRSIG5 is to exploit a more efficient and uniform multiple-scatter radiometry architecture while simultaneously easing the burden on users to appropriately setup a simulation. To accomplish this, the individual radiometry solvers (highly configurable, per-material leaving radiance algorithms in DIRSIG4) have been replaced with a unified path-tracing[3] based solution. This approach allows for a single fidelity knob (the number of samples into what is essentially a set of Monte Carlo integrals) and near uniformity in computational units (paths) to facilitate parallelization. Optical properties are still defined spectrally, but with mechanisms to handle dispersion and a single high-performance data-driven model for all but the most trivial reflectance and scattering distributions.

Outside of the core radiometry engine improvements, the most significant change in DIRSIG5 is the availability and use of application program interfaces (APIs) for a number of simulation components. Each of the APIs are documented, detailed interfaces for passing information in and out of the core. Some of these APIs are designed to be exposed publicly in order to satisfy user needs that may not be otherwise met. One such API is the sensor API, which is discussed for the remainder of the paper. It is unique in that one or more sensor plugins drive the entire simulation by interacting with the internal radiometry calculation scheduler.

2. SENSOR MODEL API

The API for sensor models in DIRSIG5 provides the basic interface between the radiometry core, the scheduler and one or more sensors being modeled during a given simulation. The API is callback driven, with the scheduler initiating a series of
calls to the sensor plugin to acquire data about the radiometry calculations to be performed and notifying the plugin when those calculations have completed and can be used.

The first call into a sensor model plugin is an initialization function, which is called by the scheduler at the start of the simulation and allows the plugin to read in external data (such as a file describing the sensor) and construct any internal data structures that it will utilize during the simulation. The model will then ask the plugin to describe a one or more “tasks”, which are time windows during which the sensor will be collecting data. Next the model will ask the plugin to describe one or more “captures” within each task. A capture is an event during which a set of radiometric “problems” (an accumulation of light transport paths) are computed by the radiometry core. To equate these abstract task and capture concepts into something practical, consider a video camera with a 2-D array of pixels that integrate for some period of time and are read out on a regular interval. A task is a time window during which the camera is on. The camera focal plane is read out at a regular interval (gross time scale), with each read out being a capture. Each capture is composed of N pixel problems, which span the integration time window (fine time scale).

Once the scheduler has acquired the task and capture descriptions and constructed it’s internal structures to efficiently execute the required radiometry calculations given the available computing resources, it will start the simulation. The sensor plugin will be notified that a task has started (identified by a task index) and then notified that a capture has started (identified by a task and capture index). The sensor plugin will then be requested to describe a specific problem identified by a unique problem index, capture index and task index. A problem description includes several important pieces of data including the radiometric convergence criteria and a pointer to a callback function within the plugin that will generate the pixel sampling rays. The radiometry core will then call the supplied ray generation function to initiate paths in the path tracer and accumulate radiance paths until the supplied convergence criteria is met. The result of a completed pixel is a spectral radiance vector. On a regular interval, the sensor plugin will be asked to process those resulting spectral radiances for each pixel problem. This is where the plugin can incorporate additional spectral contributions and/or convert the spectral radiance into an output signal.

3. SENSOR MODEL IMPLEMENTATION

Many remote sensing systems are composed of multiple instruments (cameras) and each instrument can contain multiple focal planes. For example, the Landsat-8 vehicle contains separate reflective- and thermal-region imaging systems and UAVs might have multiple cameras mounted in a single commandable ball. The “BasicPlatform” sensor plugin included in DIRSIG5 implements the multi-instrument system model introduced in DIRSIG4. That model establishes a top-level, platform (vehicle) coordinate system within which the user can define one or more instruments at various platform-relative locations and orientations. Each imaging instrument defines an internal camera geometry model using a description of the various focal planes (e.g. pixel counts, pixel sizes, pixel spacing, etc.) contained in the instrument. To generate a pixel sample ray, a location within the pixel area and the effective focal point are used. The resulting ideal pin-hole camera ray can then be modified by an optional lens distortion model. This final pixel ray is then transformed from the instrument coordinate system to the platform coordinate system using the series of transforms positioning the instrument relative to the platform. Once the pixel ray is in the platform coordinate system, it is then transformed into the scene coordinate system using the platform motion model, which defines the platform’s location and orientation as a function of time.

In terms of platform location and orientation, the “programmed motion” options in the plugin include a set of data-driven mechanisms (e.g. location and orientation as a function of time in either the scene or a geolocated coordinate system), parametric models (e.g. altitude, heading and speed) and algorithmic models (e.g. an SGP4 orbital propagator combined with orientation constraint models). The “unprogrammed motion” or “jitter” can be independently defined for all six degrees of freedom at the platform level (XYZ location and XYZ orientation) and two degrees of freedom at the platform-relative mount level (along-track and across-track pointing) using either a temporally uncorrelated model (normally distributed deviates from a user-supplied mean and standard deviation) or a temporally correlated model driven by a user-supplied power spectral density (PSD).

Each focal plane can have a unique readout rate defined by a clocking description that allows for asynchronous clocks (independent rate), synchronous clocks (multiple clocks slaved to a master clock in the platform via dividers and offset counters) and irregular clocks (a user-supplied file containing an arbitrary sequence of trigger times).

A variety of spatial and temporal camera effects can be accomplished in the generation of rays produced by the pixel sampling callback function. A point spread function (PSF) externally generated by an optical modeling package (e.g. Zmax, Code V, etc.) can be imported as a 2-D image. The convolution of the detector area and PSF is then accomplished by a two-step sampling process to generate the ray origin. For a given ray, a random location within the pixel area is computed and then the PSF map is importance sampled to offset that ray origin. Hence, an ideal PSF (delta function) produces no offset and a realistic PSF will produce a distribution of sampling inside and outside the geometric instantaneous field-of-view (IFOV) of the pixel. Temporal integration can be accomplished by distributing the time associated with individual sample rays across the integration period (fine time scale). A rolling shutter can be accomplished by offsetting
those ray times based on the line (row) of the array the pixel is in. Since there are temporal components (platform-relative pointing, jitter, etc.) in the chain of operations that transform instrument relative rays to the scene coordinate system, the radiance onto a pixel from a large enough set of sampling rays will directly incorporate modulation transfer function (MTF) effects due to fine scale motion (e.g., jitter induced blur) and course scale motion (e.g., along-track ground sample distance (GSD) elongation).

In order to make this complex, multi-instrument platform hierarchy model to the simple interfaces defined by the sensor API, the plugin manages an internal table to map individual captures supplied to the scheduler to a specific instrument and focal plane. The set of captures to be executed by the radiometry core for a given task are generated using the approach illustrated in Figure 1. The plugin traverses the instrument hierarchy and has each focal plane clock generate capture events that occur during the supplied task time window. Those capture events are stored in an internal mapping table that include the indexes back to the instrument and focal plane within that instrument for the capture. Once all the captures are collected across all the instruments, the internal map is temporally sorted. Although the API doesn’t require the captures to be ordered in time, the user expectation is that data will be generated sequentially in time. The sorted mapping table is then used to fill the radiometry scheduler with a series of captures through the capture query callbacks.

Once the simulation is started, the sensor plugin will need to react to a sequence of callbacks that enable the radiometry core to compute the spectral radiance products required by the sensor plugin. This execution flow is illustrated in Figure 2 including how the internal mapping table is used at the start of the capture to determine which focal plane in which instrument is about to be modeled.

![Scheduler](image1.png)
![BasicPlatform Sensor Plugin](image2.png)

**Fig. 1.** Generating the schedule of external captures events and internal tables to resolve those events.

**Fig. 2.** The flow of the API callbacks for a single capture.

### 4. SENSOR MODEL DEMONSTRATIONS

To demonstrate some of the capabilities of the “BasicPlatform” plugin, a series of simulations are provided that highlight common effects and artifacts observed in remotely sensed imagery. These simulations were performed on a scene constructed of a shipping dock at the Port of Tacoma, WA, USA. The scene was created using overhead imagery to drive the terrain material types and variability. Onto this terrain were placed an array of 3D objects including a ship, cranes, containers, buildings, various vehicles and a set of calibration targets. The scene contains a variety of dynamic objects including spinning rotors on a helicopter, driving vehicles, moving boats and walking humans. A panchromatic, overview simulation of the scene is shown in Figure 3. The helicopter and calibration target area in the lower-right corner of the image will be utilized in the following demonstrations.

![DIRSIG simulation of the Port of Tacoma, WA scene.](image3.png)

Many airborne remote sensing systems utilize 2-D framing detector arrays to image the scene. The focal planes in these systems might utilize a global shutter, where all the pixels are integrated and then read out, or a rolling shutter, where lines in the array are independently integrated and then read out sequentially. The advantage of the later is that pixels can be integrated for a greater portion of the time between read
outs because they continue to integrate while other pixels are being read out. However, a rolling shutter produces a unique set of artifacts when either the scene, sensor or both are moving. The simulated images of the helicopter with spinning rotor blades below demonstrate some of these effects. In Figure 4(a), the focal plane is emulating a global shutter scenario and the motion blur of the blades can be observed. In Figure 4(b), a rolling shutter scenario is modeled and the common artifact of the rotor blades “bending” can be observed. These effects are accomplished by sampling each pixel in the spatial and temporal domains and not with pre-processing of the scene or post-processing of the images.

![Global shutter](image1.png) ![Rolling shutter](image2.png)

**Fig. 4.** The impact of global vs. rolling shutter read out.

Many large-area, space-based remote sensing systems utilize a pushbroom scanner focal plane architecture (e.g. IKONOS, WorldView 2/3/4, etc.), where linear, across-track arrays of pixels (effectively, N x 1 arrays) scan the scene in the along-track direction using vehicle motion and/or vehicle pointing. With limited time over the scene (due to the orbit), the along-track sampling (resolution) and integration time must be balanced. However, the signal-to-noise (SNR) is primarily limited by the integration time and if the integration time is increased too much, the effective GSD of the system increases in the along-track direction. A series of collection scenarios over the tri-bar and calibration targets are presented below to illustrate the trade space between resolution and noise (note: the SNR is artificially high for visualization purposes). The system is orbiting at a nominal altitude of 585 km, with an effective ground speed of 7.6 km/sec and the focal plane is read out at 25 kHz to yield a 0.3 meter GSD. The simulations utilize the same spatial and temporal pixel sampling mechanisms discussed previously and do not utilize any post-processing to blur or rectify the images. In Figure 5(a), the system uses the entire time between array readouts to integrate. In Figure 5(b), the integration time was increased 8x, which results in an improvement in the SNR, but the image is blurred in the along-track direction. The image in 5(c) is from a focal plane that is N x 8 pixels, where the 8 pixels in the along-track direction are used for time-delayed integration (TDI) to achieve a similar SNR improvement. TDI allows the system to mitigate the noise through an 8x longer effective integration while preserving along-track image quality. This is modeled by distributing samples in space and time across the 8 along-track TDI stages.

![Short integration](image3.png) ![Long integration](image4.png) ![Using TDI](image5.png)

**Fig. 5.** The impact of various temporal integration techniques.

5. SUMMARY

The sensors described through the sensor plugin API are the primary driver of a DIRSIG5 simulation. The bundled “BasicPlatform” sensor plugin has been shown to effectively employ spatial and temporal sampling techniques to model a variety of image features and artifacts frequently observed in remote sensing systems. The use of these sampling techniques has been found to facilitate easy to use data-driven interfaces and avoids sometimes difficult image processing approaches. Although not discussed here, the same sensor API can be used to model non-imaging systems including arrays of point radiometers and goniometers.

6. REFERENCES

