Open Geospatial Consortium, Inc.

Date: 2008-09-12

Reference number of this document: OGC 08-071

Version: 0.1.0

Category: Discussion Paper

Editor(s): Mike Botts

OWS 5 Engineering Report: Supporting Georeferenceable Imagery

Warning

This document is not an OGC Standard. This document is an OGC Discussion Paper and is therefore not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, an OGC Discussion Paper should not be referenced as required or mandatory technology in procurements.

Preface

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

Forward

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Contents

Page

Supporting Georeferenceable Imagery

1 Introduction

1.1 Scope

The scope of this document is to capture considerations and recommendations on approaches for supporting georeferenceable imagery within the OGC encodings and web services. Georeferenceable imagery is typically imagery coming from a remote sensor that has not been previously geo-rectified, resampled, or regridded. Georeferenceable imagery must be accompanied with information sufficient to allow georectification of the imagery. Out of scope for this document are *in situ* sensors where the location of the observed phenomenon is considered to be a point that is coincident with the location of the sensor itself.

This document explores the possible roles of OGC Sensor Web Enablement (SWE) encodings [i.e. Sensor Model Language (SensorML), Transducer Markup Language (TML), and Observations and Measurements (O&M)], as well as OGC web services, such as Sensor Observation Service (SOS), Web Coverage Service (WCS), and Web Feature Service (WFS).

1.2 Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Mike Botts	University of Alabama in Huntsville

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
12/07/2007	1.0.0	botts		initial document begun
08/11/2008	1.0.0	botts		Completion of V1.0 submission version

1.4 Future work

Improvements in this document are desirable to provide more concrete examples of the use of OGC standards to support Georeferenceable Imagery. In addition, as standard sensor models are refined and completed, this document should be updated to reflect those changes.

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

OGC 00-116, *The OpenGIS Abstract Specification, Topic 16: Image Coordinate Transformation Services,* Version 6

OGC 03-105r1, *OpenGIS Geography Markup Language (GML) Encoding Specification*, Version 3.1.1

OGC 05-008, *OGC Web Services Common Specification,* Version 1.0.0

OGC 05-047r3, *OpenGIS GML in JPEG 2000 for Geographic Imagery Encoding Specification*

OGC 05-096r1, *GML 3.1.1 grid CRSs profile*

OGC 06-009r5, *Sensor Observation Service*

OGC 06-083r8, *OpenGIS Web Coverage Service (WCS) Implementation Specification,* Version 1.1.0

OGC 06-111, *GML 3.1.1 grid CRSs Profile Corrigendum*

OGC 06-121r3, *OpenGIS® Web Services Common Specification*

OGC 05-099r2, *GML 3.1.1 simple dictionary profile*

OGC 05-103, *The OpenGIS Abstract Specification, Topic 2: Spatial Referencing by Coordinates*

OGC 06-010r6, *Transducer Markup Language (TML) Implementation specification*, Version 1.0.0.

OGC 07-000, *OpenGIS Sensor Model Language (SensorML),* Version 1.0.0

OGC 07-002r3, *Observation and Measurements – Part 2 – Sampling Features*

OGC 07-006r1, *OpenGIS Catalog Service Implementation Specification*

OGC 07-022r1, *Observation and Measurements – Part 1 – Observation Schema*

OGC 07-030r1, *OpenGIS Image Geopositioning Service (IGS)*

OGC 07-031r1, *OpenGIS Image Geopositioning Metadata GML 3.2 application schema*

OGC 07-036, *OpenGIS Geography Markup Language (GML) Encoding Specification*, Version 3.2.1

OGC 07-055, *Web Coordinate Transformation Service (WCTS) draft Implementation Specification*

OGC 07-067r2, *OpenGIS Web Coverage Service (WCS) Implementation Specification Corrigendum 1 (1.1.1 c1)*

OGC 07-112, *GML 3.2.1 CR – Add implementation of ISO 19123 CV_ReferenceableGrid to GML*

ISO 19101:2002. *Geographic information – Reference Model*

ISO 19109:2004. *Geographic information - Rules for application schema*

ISO 19111:2007. *Geographic information – Spatial Referencing by Coordinates*

ISO 19115:2003. *Geographic information – Metadata*

ISO 19115:2006. *Geographic information – Metadata, Technical Corrigendum 1*

ISO CD 19115-2:2005. *Geographic information – Metadata – Part 2: Metadata for imagery and gridded data*

ISO 19123:2005. *Geographic information – Schema for coverage geometry and functions*

ISO CD 19130.1.0:2007. *Geographic information – Imagery Sensor Models for Geopositioning*

ISO DTS 19139:2005. *Geographic information – Metadata – XML Schema Implementation*

In addition to this document, this report includes several XML Schema Document files as specified in Annex A.

3 Terms and definitions

3.1 attitude

orientation of a body, described by the angles between the axes of that body's **coordinate system** and the axes of an external **coordinate system** [ISO 19116]

3.2 coordinate
one of a sequence of *n* numbers designating the position of a point in *n*-dimensional space [ISO 19111]

3.3 coordinate reference system
coordinate system that is related to the real world by a datum [ISO 19111]

3.4 coordinate system
set of mathematical rules for specifying how **coordinates** are to be assigned to points [ISO 19111]

3.5 datum
Undefined in ISO 19111. Defined here as a means of relating a **coordinate system** to the real world by specifying the physical location of the coordinate system and the orientation of the axes relative to the physical object. For a geodetic datum, the definition also includes a reference ellipsoid that approximate the surface of the planetary body.

3.6 detector
Atomic part of a composite **Measurement System** defining sampling and response characteristic of a simple detection device. A detector has only one input and one output, both being scalar quantities. More complex **Sensors**, such as a frame camera, which are composed of multiple detectors can be described as a detector group or array using a **system** or **sensor**.

3.7 frame sensor/frame camera
sensor that detects and collects all of the data for an **image** (frame / rectangle) at an instant of time [ISO 19130]

3.8 functional fitting model
functional relationship between ground and **image coordinates** based on the correlation between a set of **ground control points** and their corresponding **image coordinates**. [ISO 19130]

3.9 georectified image
an image that has undergone the process of **georectification**

3.10 georectification

3.11 georeferenceable image

3.12 georeferencing transformation
coordinate transformation that can be used to convert grid coordinate values to values of coordinates referenced to a coordinate reference system that is related to the earth by a datum (adapted from ISO 19123)

3.13 image
coverage whose **attribute** values are a numerical representation of a remotely sensed physical parameter [ISO 19130]

3.14 image coordinates
coordinates with respect to a Cartesian coordinate system of an **image** [ISO 19130]

3.15 location
A point or extent in space relative to a **coordinate system**. For point-based systems, this is typical expressed as a set of n-dimensional coordinates within the **coordinate system**. For bodies, this is typically expressed by relating the translation of the origin of an object's local **coordinate system** with respect to the origin of an external reference **coordinate system**.

3.16 location model
A model that allows one to locate objects in one local **reference frame** relative to another **reference frame**.

3.17 measurand
Physical parameter or a characteristic of a phenomenon subject to a **measurement,** whose value is described using a Measure (ISO 19103). Subset of **determinand** or **observable**. *[O&M]*

3.18 measure (noun)
Value described using a numeric amount with a scale or using a scalar reference system *[ISO/TS 19103]*. When used as a noun, measure is a synonym for physical quantity.

3.19 measurement (noun)
An **observation** whose result is a **measure** *[O&M]*

3.20 measurement (verb)
An instance of a procedure to estimate the value of a natural phenomenon, typically involving an instrument or sensor. This is implemented as a dynamic feature type, which has a property containing the result of the measurement. The measurement feature also has a location, time, and reference to the method used to determine the value. A measurement feature effectively binds a value to a location and to a method or instrument.

3.21 observation

3.22 orientation
The rotational relationship of an object relative to a coordinate system. Typically expressed by relating the rotation of an object's local coordinate system relative to an external reference coordinate system.

3.23 orthorectified image
an image that has undergone the process of **orthorectification**

3.24 orthorectification
a process that removes geolocation errors arising from oblique look angles and the heights of various objects in the scene

3.25 pixel
picture element [ISO 19101-2]

3.26 (sensor) platform
An entity to which can be attached sensors or other platforms. A platform has an associated local coordinate frame that can be referenced to an external coordinate reference frame and to which the frames of attached sensors and platforms can be referenced.

3.27 position
The location and orientation of an object relative to a coordinate system. For body-based systems (in lieu of point-based systems) is typically expressed by relating the object's local coordinate system to an external reference coordinate system. This definition is in contrast to some definitions (e.g. ISO 19107) which equate position to location.

3.28 process
A process that takes one or more inputs, and based on parameters and methodologies, generates one or more outputs.

3.29 process chain
Composite processing block consisting of interconnected sub-processes, which can in turn be Process Models or Process Chains. A process chain also includes possible data sources as well as connections that explicitly link input and output signals of sub-processes together. It also precisely defines its own inputs, outputs and parameters.

3.30 rational polynomial coefficients (RPC)

3.31 reference frame
A coordinate system by which the position (location and orientation) of an object can be referenced.

3.32 remote sensing
collection and interpretation of information about an object without being in physical contact with the object

NOTE: typically resulting from the collection and measurement of reflected or emitted electromagnetic radiation from a feature of interest

3.33 replacement sensor model

3.34 rigorous sensor model / physical sensor model
model that uses the physical configuration of **sensor** to mathematically derive the geometric relationship between the location of a **pixel** in the two-dimensional **image plane** and its location in the three-dimensional object space

3.35 Sample
A subset of the physical entity on which an observation is made.

3.36 Sensor
An entity capable of observing a phenomenon and returning an observed value. In SensorML, modeled as a specific type of **System** representing a complete **Sensor**. This could be for example a complete airborne scanner which includes several **Detectors** (one for each band).

3.37 Sensor Model
In line with traditional definitions of the remote sensing community, a sensor model is a type of Location Model that allows one to georegister observations from a sensor (particularly remote sensors).

3.38 sensor
element of a measuring instrument or measuring chain that is directly affected by the measurand [ISO 19101-2]

3.39 sensor model
mathematical description of the relationship between the three-dimensional object space and the associated two-dimensional **image plane** [ISO 19130]

defined method and associated parameters required to reference observations to a desired coordinate reference system

4 Conventions

4.1 Abbreviated terms

API	Application Program Interface
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off The Shelf
CRS	Coordinate Reference System
CSM	Community Sensor Model

DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
EPSG	European Petroleum Survey Group
GML	Geography Markup Language
IDL	Interface Definition Language
ISO	International Organization for Standardization
OGC	Open Geospatial Consortium
OWS	OGC Web Service, or Open Web Service
RPC	Rapid positioning Coordinates / Rational Polynomial Coefficients
SensorML	Sensor Model Language
SAS	Sensor Alert Service
SOS	Sensor Observation Service
SWE	Sensor Web Enablement
TML	Transducer Markup Language
UAH	University of Alabama in Huntsville
URI	Universal Resource Identifier
URL	Uniform Resource Locator
URN	Universal Resource Name
WCS	Web Coverage Service
WFS	Web Feature Service
XML	Extensible Markup Language
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional

4.2 UML notation

A few diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 06-121r3].

5 Overview

5.1 Remote Sensing and Georeferenceable Imagery

Georeferenceable imagery is typically imagery coming from a remote sensor, in which the observation has not previously been georectified, resampled, or regridded. Georeferenceable imagery must be accompanied with information sufficient to allow georectification of the imagery upon its receipt. This is in contrast to *in situ* sensors where the location of the observed phenomenon is considered to be a point that is coincident with the location of the sensor itself, and in contrast to georectified imagery in which the observations have been resampled and mapped into a geospatial domain.

A remote sensor is one that typically collects and measures the properties of electromagnetic radiation that has been reflected or emitted from a distant feature of interest. This radiation is often transmitted through a medium (e.g. the atmosphere or ocean) where its strength might be attenuated relative to the spectral frequencies of the radiation and the presence of objects within the medium. The remote sensor usually involves a system that collects radiation and directs it to a detector or detector array. The collection system might consist of a set of lenses, prisms, and mirrors for electromagnetic radiation between UV and IR wavelengths or one or more conic dishes for the collection microwave radiation. Other collection devices may exist for other frequencies (e.g. acoustic, seismic, etc.), but these sensors are currently out of scope for this document.

The detector system of a remote sensor can consist of a single “chip” or detector, or of a regularly spaced array of detectors. The collection system can be designed such that radiation of different frequencies impinge upon different detectors, thus allowing one to measure properties of the radiation (e.g. intensity or phase) for separate frequency bands. For example, multi-spectral sensors might measure the intensity of the radiation within 2-20 individual frequency bands, whereas hyper-spectral sensors might take individual measurements for 64-256 bands.

In order to determine the precise location of the objects of interest with a given remotely sensed scene, one must utilize models (typically referred to as “sensor models”) that provide a mapping of positions in the image to positions within the environment. The design of the collection system and its relationship to the detector and platform on which it is attached, as well as the dynamics of the system, generally dictate a rigorous physical model for this mapping.

Many of these remote sensors generate a spatial-temporal coverage of observation points that can be regularly arranged as a grid and viewed as an image. While the term “observation coverage” is more general and more appropriate for describing the output of a remote sensor, the use of the term “imagery” is heavily engrained in the remote sensing community and has a long history beginning with the use of film for capturing electromagnetic radiation. Thus, within this text, the term image will generally be considered synonymous with a “regularly” gridded collection or coverage of observations coming from a remote sensor.

One is cautioned, however, that remotely sensed observations can include coverages, such as profiles, that are not what one would typically consider imagery. Furthermore, use of the term “imagery” often taints a proper understanding of the sampling and georeferencing of

observations from a remote sensor. For example, consider the case of multi-spectral remote sensor on board a polar-orbiting satellite, in which the sensor continuously scans side-to-side (i.e. cross-track) while the satellite continuously moves along an orbital path (i.e. along-track). In the purest conceptual thinking, the satellite sensor system is simply producing a time series collection of measurements. It is the temporal-spatial nature of the sampling that suggests that one could represent the coverage as an image, but one could just as easily be only interested in the spectral curve on a given pixel. [see Discussion on the Nature of Observations in Appendix A]. In either case, the ability to georeference the sample to a geospatial location using a sensor model is usually required for the observation to be of any use.

While georeferenceable imagery typically exists within a regular grid, this grid is not regular within any geodetic coordinate system. That is, the imagery is usually obliquely oriented to geodetic systems (e.g. the latitude-longitude grid) and one cannot expect the spacing of pixels in the imagery to be equal distance; nor can one expect the size of the area covered by each pixel to be equal in area. While some previous data suppliers have provided their clients with pre-calculated latitude, longitude, and perhaps altitude for each pixel in the scene, this approach often results in larger file sizes, higher band width requirements, inflexibility with regard to correcting errors or mapping to various surfaces, and increased delays in receiving the data.

5.2 The Importance of Supporting Georeferenceable Imagery

Within various remote sensing communities, including earth observation, resource management, defense, and intelligence, support for Georeferenceable imagery has, until recently, been primarily an issue for large data centers. These centers would typically georectify such imagery products and in many cases resample and regrid these data into products that were easily consumed by their customers. While there is often still a need for such processing at data centers, there are many advantages to providing support for the processing of georeferenceable imagery further down the data pipeline, including on-demand georectification within the user's local decision support tool.

Furthermore, data required for precise geolocation of remotely sensed data has typically not been provided to end users or has been treated as metadata or Ancillary data associated with a particular data set. These data include, for instance, platform locations and attitude, sensor gimbal rotations and positioning, sensor status, and internal sensor parameters. It is important that these data begin to be viewed as actual data coming from the platforms and its attached sensors, and not simply metadata stored away in a data center or only attached to a particular image as metadata. For example, being able to retrieve subsetted platform position, platform attitude, gimbal rotations, and sensor status independent of any particular observation is critical to improving the rapid discovery of archived or real-time georeferenceable imagery that might meet particular time and position requirements.

Briefly, some of the reasons for supporting the geolocation or georectification of imagery at the client or within a workflow, can be categorized as:

- **discovery** – as discussed above, the ability to retrieve ancillary data independent of the data itself, combined with the ability for on-demand geolocation of sensor enables significant improvements in the discovery of available remotely sensed observations

- **reduction of latency in real-time imagery** – by enabling geolocation to be supported further down the data pipeline (perhaps at the client), one is not dependent on the scheduling and efficiency of data processing at the data center; in essence, observations can be immediately transmitted to the client upon ingestion at the data center, and in many cases, can be transmitted directly to the client from the sensor. Combined with technologies such as JPIP tile streaming, it was demonstrated in OGC OWS 5.1 that very large (40 GB) georeferenceable imagery from high-resolution sensors could immediately be accessed and explored by clients that are capable of on-demand geolocation (e.g. a SensorML-enabled client).

- **decrease in data volume and bandwidth** – for a single band data set, the inclusion of pre-calculated latitude, longitude, and altitude can increase the file size and band-width requirements, by 4 times or more, depending on the need for other pre-calculated values which could instead be provided by capabilities at the client (e.g. geolocation accuracy values, smear factors, incidence angles, sun angles, etc.).

- **flexibility of geolocation** – typically at a data center, pre-processed data products are generated to meet the average or largest user of the data. Once a product has been geolocated and perhaps aggregated relative to a particular surface (e.g. digital terrain model, ellipsoid, sea-surface, etc.), the client is not able to reposition the data based on other surfaces or to calculate other parameters that might be needed. For example, geolocating atmospheric events relative to an ellipsoid or terrain can significantly misrepresent the position of that object or phenomenon.

- **flexibility of aggregation and products** - by providing unrectified georeferenceable imagery along with the ability for on-demand precise geolocation, one provides maximum flexibility with regard to both spatial and temporal aggregation of data values. The processes of rectification and aggregation are usually irreversible if one desires a projection or time resolution that is different than the one provided.

- **correction and adjustment** – for georectified imagery, often one is not aware of inaccuracies in geolocation until one has received and begun dealing with the data. While is it possible in many cases to correct some geolocation errors using tie points and "image stretching", these techniques are not based on any understanding of the sensor model. With georeferenceable imagery, it is often possible to test and adjust parameters with known inaccuracies (e.g. aircraft pitch-roll-yaw) and to thereby correct the geolocation using rigorous models.

5.3 Overview of the Georeferencing Process

The process of precise geolocation or georeferencing of remotely sensed observations involves rigorous models which account for the sensor's method of collecting the radiation and directing it to the detectors, as well as modeling the relative dynamic position, orientation, and velocity between the sensing system and the observed feature or features of interest. Which model is used to georeference a given set of observations will depend on the type of sensor, the characteristics of the dynamic or stationary platform, and the nature of the observed feature of interest.

Models for precise geolocation can be classified as rigorous models or functional fit models. **Rigorous models** attempt to mathematically reflect the physical nature of the sensor system, as well as the physical relationships between the sensor system, its platform, and the observed space. **Functional fit models** provide a mathematical transform between image space and geodetic space in which the physical nature of the relationship is no longer maintained. The functional fit model is typically derived from a rigorous model, but no longer reflects the physics of the components. In either case, a model defines:

- the expected inputs and outputs
- the parameters needed (e.g. sensor characteristics, platform positioning, etc.)
- the mathematical algorithm

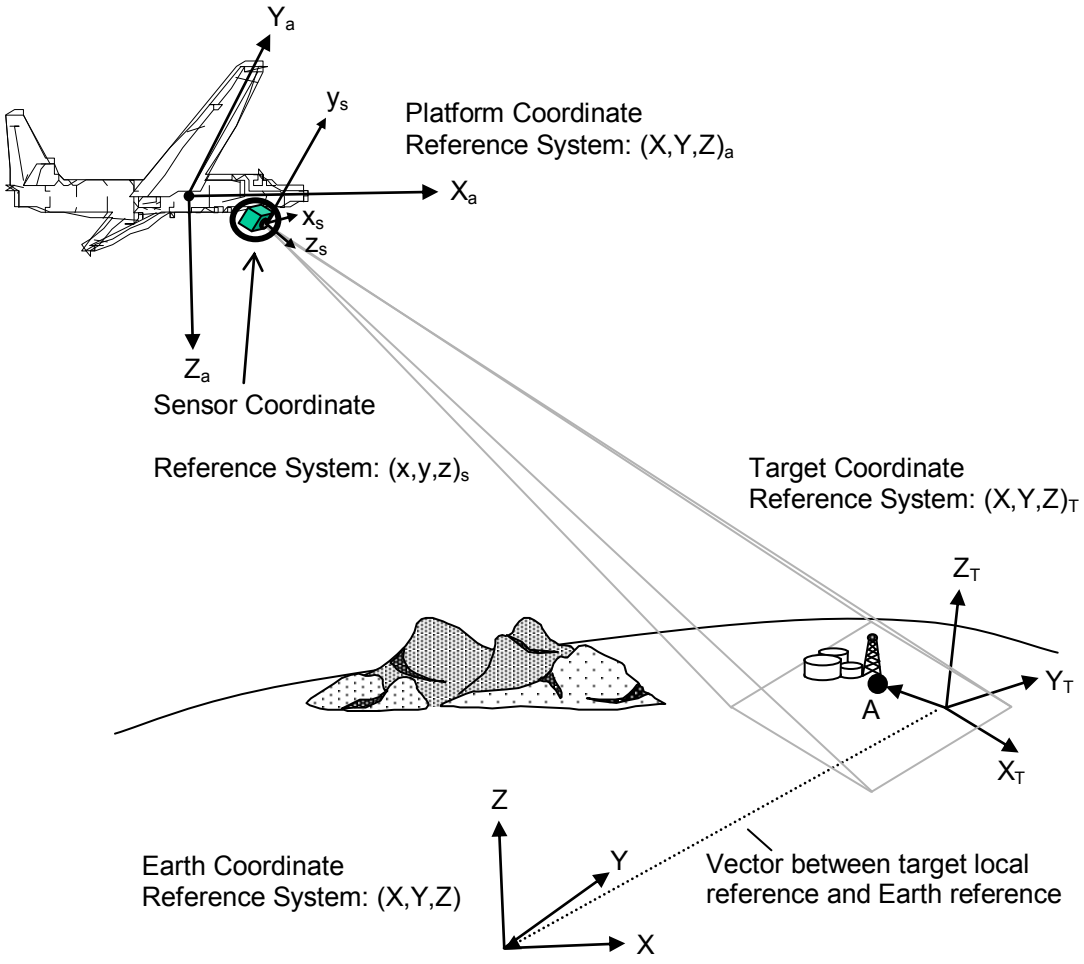

Figure 1. Multiple coordinate reference frames with a Frame Sensor (source: CSM_Frame Sensor Model document [CSM1])

Figure 1 illustrates an imaging sensor mounted on an airborne platform observing a feature of interest on the Earth's surface. The challenge of precise geolocation is to determine that a given pixel represents a particular well-defined area on the surface. This process must be able to account for several components:

- a mathematical description of the radiation collection system of the sensor and its relationship with the sensor's detection system
- the precise measurement of the position, orientation, and dynamics of the platform
- the precise measurement of the relative position and orientation of the sensing system relative to the platform, as well as that of any gimbals to which the sensor might be attached
- precise mathematical representation of the position and shape of the surface or volume being observed (accounting perhaps for ground terrain and man-made structures)
- to some extent, an understanding of the nature of the reflected and/or transmitted radiation emanating from the observed object
- the potential effects (e.g. attenuation and bending of light) resulting from the transmission of the observed radiation through the media existing between the sensor and the observed feature

Thus, a given model will define the parameters needed to adequately describe the properties of the components listed above, as well as the algorithm that will use these parameters and mathematically define the real the relationship between a given observation sample and its position in geodetic space.

In its simplest form, the relationship between a particular element on a detector array and the associated observed location on the ground can be modeled as a "look ray" that extends between the observed object and the affected element of the detector array. A slightly more precise representation might utilize a conic volume for imagers or a lobed volume for radar. Regardless, the precise geolocation model must deal with transformations between several reference system, a few of which are illustrated in Figure 1.

Typically models exist to allow transformations in both from ground position to image position (ground-to-image) and from image position to ground position (image-to-ground). Depending on the type of remote sensor and the nature of the observed feature being geolocated, each method has certain advantages and disadvantages. For example, ground-to-image functions work well when one must account for an irregularly shaped terrain observed by a frame camera, but can be more challenging when dealing with scanners. Often, however, the parameters required for a given sensor system will be the same, regardless of whether one uses an image-to-ground or ground-to-image model.

Finally, rigorous models often tend to separate the model of the internal working of the sensor (e.g. the collection and detector system) from the model of the external influences (e.g. platform dynamics, medium characteristics, and observed feature characteristics). Thus, the following sections will separate the platform descriptions and models from those of the sensor. This approach allows better reuse and modularization of models, when "mixing and matching" platform types with sensor types. In other words, the sensor model for a frame camera should

not need to change regardless of whether the sensor is mounted on a satellite, aircraft, or at a fixed station on the ground. However, the influence of the platform type can be separately accounted for by the use of an appropriate platform model.

6 Types of Remote Sensors

Sensors can be classified based on several criteria, including:

- in-situ versus remote sensor
- static versus dynamic platform
- platform type (e.g. satellite, airborne, ground-based, ship-based, etc.)
- electromagnetic sensitivity (e.g. infrared, visible, acoustic, etc.)
- active versus passive
- sampling type (e.g. frame sensor, whiskbroom scanner, profiler, etc.)

These classifications are not necessary independent of one another. For example, satellite and aircraft are obviously dynamic platforms, while some ground-based platforms can be dynamic or static. In some cases, sensors of a particular electromagnetic frequency range (e.g. radar) typically utilize particular sampling methods.

However, for the scope of this document, we will primarily focus on remote sensors on both static and dynamic platforms, and will typically focus on differences in the sampling types. The platform type will become important only because certain platforms may use particular methods for determining or recording position and attitude. For example, satellite position can be determined using parameters describing its particular orbit or using **Global Positioning System (GPS)** positions.

6.1 Sampling

A rigorous sensor model must reflect the geometry and dynamics of the sampling system of the sensor. The sampling system in a remote sensor typically consists of a collection of lenses, mirrors, and filters that gather radiation of particular spectral wavelengths and focus that radiation unto one or more radiation sensitive detector chips. The geometric arrangement of the collection system and detectors, coupled with the dynamics of the platform and sensor system itself must be accounted for by the sensor model.

6.1.1 Frame Camera/ Frame Sensor

A **frame camera**, or **frame sensor** is a sensor that acquires all of the data for an image (frame) at a single instance of time [ISO 19130]. The timing of multiple acquisitions can be regular or irregular. Typically, the frame sensor consist of a collection of lenses and possibly mirrors that serve to collect radiation from a targeted source and a two-dimensional detector device, such as a **focal plane array (FPA)**, a **Charge-Coupled Device (CCD)**, or film.

The output of a frame sensor is typically an image or a video stream. Thus, since the entire detector array is sampled at a single instance in time, one receives a coherent image regardless of any motion of the platform. Unlike the other sensor types described below, frame sensors do not require an intimate synchronization between the movement of the platform and the sampling of the data, although the geolocation process does depend on accurately determining the location and orientation of the platform at the time of sampling. Thus, a frame sensor can be mounted on virtually any platform, including, for example, an aircraft (as in Figure 1), a satellite, a mobile ground vehicle, a mobile water craft, or a stationary ground station).

6.1.2 Scanners – Pushbroom, Whiskbroom, Scanner/Profilers

A scanner is a sensor that builds a regular coverage of observations (e.g. an image) by measuring one or more pixels at a time. A complete coverage or image is obtained by a well-timed scanning process accomplished using rotating mirrors, gimbal rotations, or the motion of the platform itself. Scanners are classified according to their sampling behavior. The two major types include the pushbroom and whiskbroom scanners.

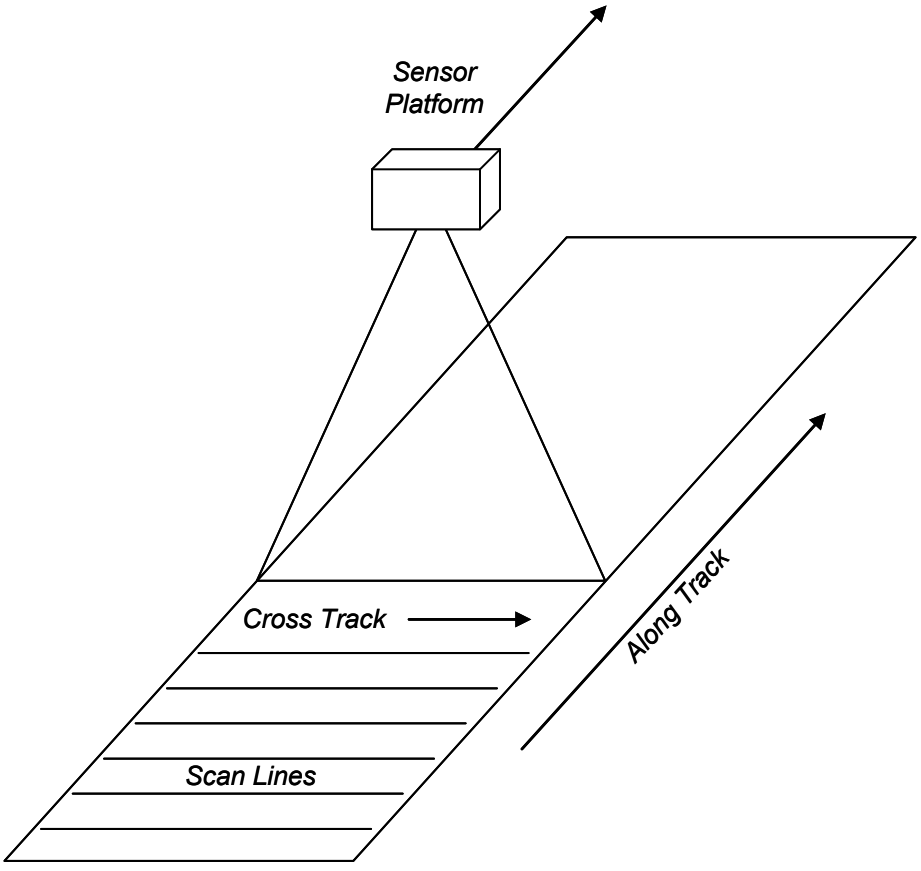

Figure 2. Simple illustration of a scanner showing the direction of scanning relative to the direction of sensor movement [ISO19130].

A **pushbroom sensor** consist of a linear array of detectors arranged in one or more rows that are perpendicular to the motion of the platform. Like a frame camera, a pushbroom sensor samples

all pixels in the array at the same time and utilizes lenses and mirrors for collecting radiation and focusing it on the detector plane. However, in a pushbroom, subsequent acquisitions are regular and are timed with the motion of the platform in order to achieve a coherent coverage of the target in the direction of platform movement. Thus, with a pushbroom sensor, samples are along the cross track direction (see Fig.2) are acquired at the same time, while scan lines are built up through time as the sensor move forward.

A **whiskbroom sensor** scans in the cross track direction while the sensor platform moves forward. A whiskbroom might sample only a single pixel at a time or a like the pushbroom several pixels at once. However, unlike the pushbroom scanner, the array of pixels is oriented in the direction of platform movement such that any samples taken synchronously are on adjoining scan lines.

As with the pushbroom scanner, the whiskbroom scanner must be intimately synchronized with the movement of the platform so that subsequently scanned rows form a more or less contiguous coverage in the direction of travel.

Whiskbroom sensors can also be classified as linear (the most common) or conic scanners. A linear scanner is oriented such that the scan line is perpendicular to the platform motion and is centered at the nadir point below the sensor (Fig. 4). While linear scanners are the most common scanner type, they have the disadvantage that the footprint size of each pixel on the ground increases as one looks further from the nadir position, due to the spheroid shape of the Earth.

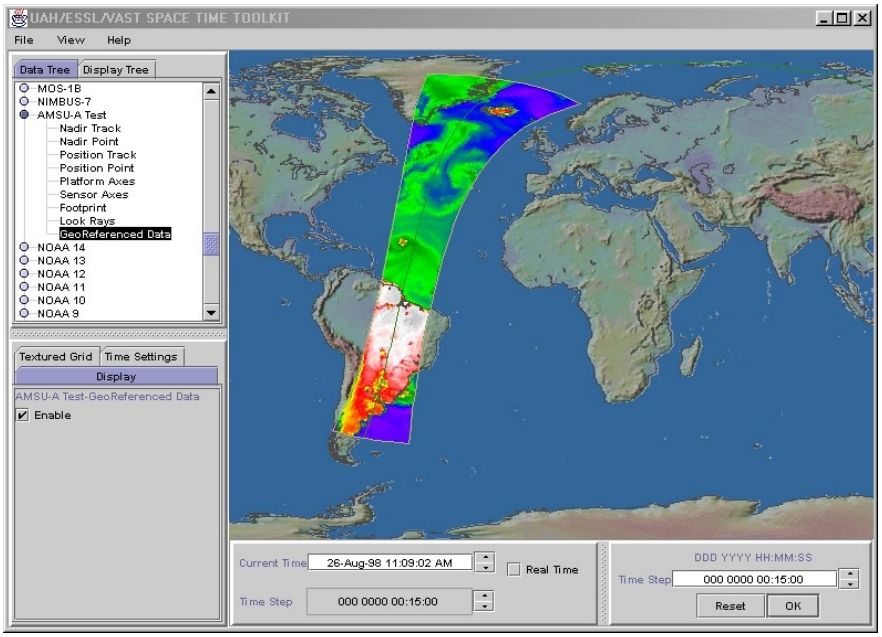

Figure 3. Georeferenced swath from a linear scanner (AMSU-A) using SensorML.

Conic scanners are instead oriented to point at some angle forward or aft of nadir, and with a scanning pattern that forms a conic-shaped footprint on the target surface (Fig. 4). The

advantage of the conic scanner is that the size of each pixel's footprint on the ground remains more constant through the scan pattern.

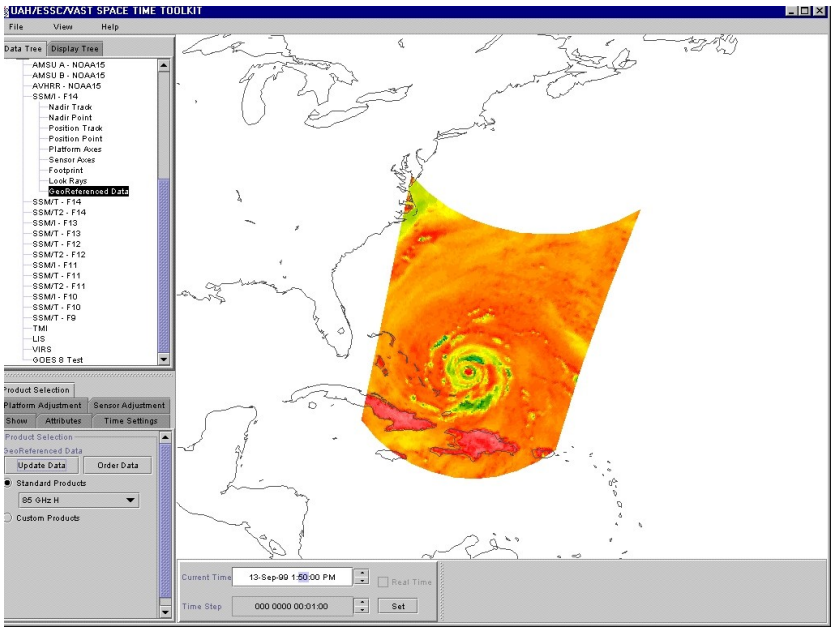

Figure 4. Georeferenced swath from a conic scanner (SSM/I) using SensorML.

Another example of a conic scanner/profiler is the ground-station mounted Doppler radar used for tracking storms (Fig. 5). The Doppler radar typically scans 360 degrees about vertical, changes elevation angle and repeats the scanning pattern creating a "volumetric" scan consisting of around 16 elevation levels.

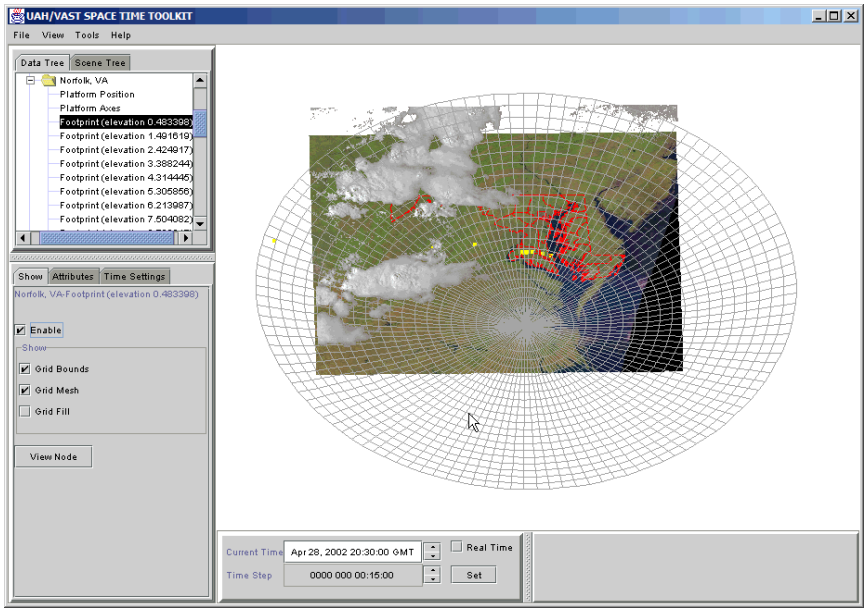

Figure 5. Georeferenced scan from a Doppler radar conic scanner/profiler using SensorML.

6.1.3 Profilers

A **profiler** is a sensor that is capable of taking measurements at various distances along the line of sight. A profiler can be designed to return measurements within several bins along the line of sight (as in Fig. 6) or simply measure the distance to the nearest feature of interest.

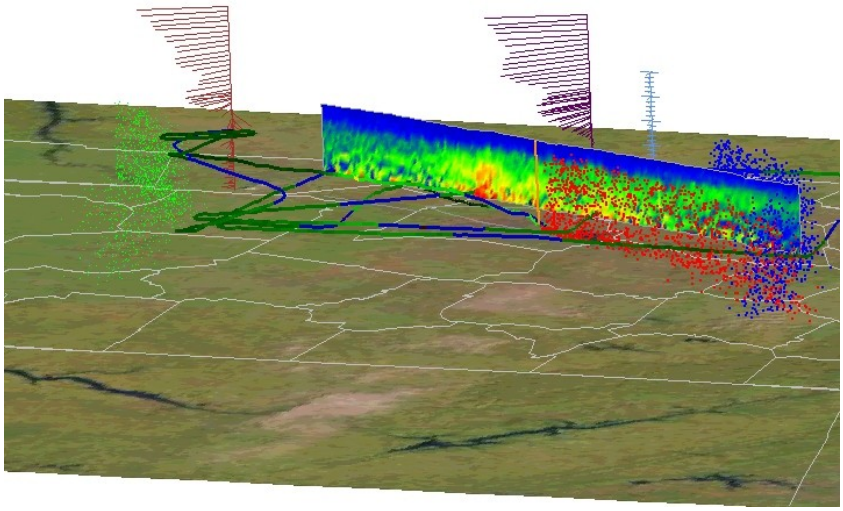

Figure 6. Georectification and visualization of two types of profilers fused with other sensor observations. There are two stationary vertical profilers that measure wind speed and direction as a function of height in the atmosphere (red and purple barbs), and a third profiler which is attached to an aircraft and measures ozone concentration along the downward line of sight (blue, green, yellow, red vertical panel).

Some sensors can provide pushbroom and whiskbroom scanning action combined with profiling. A common example of this is the Doppler radar used for observing storms. In essence, it scans in two dimensions while measuring a profile along the line-of-sight radial at each step of the scan.

6.1.4 Special - SAR, acoustic, seismic

There are other remote sensors, such as Synthetic Aperture Radar (SAR) and various acoustic and seismic sensors that are out of scope for this document. Locating the source of an acoustic or seismic signal typically relies on a dispersed, synchronized network of sensors in order to determine the direction and ultimately the distance of the source. This process can be complicated by the presence of various reflectors and refractors along the paths between the source and the detectors.

The typical SAR consists of an antenna mounted to the side of an aircraft or satellite, with the antenna often acting both as an active pulsed radar source and as a detector of the reflected signals. While the processing of low-level SAR observations (amplitude and phase) is

complicated and beyond the scope of this document, once processed, SAR observations can often be modeled frame cameras, scanners, or profilers depending on the processing of the data or its intended use.

6.2 Platform Types

6.2.1 Importance of Platform State

Sensor are generally attached to platforms which are either static or dynamic. Platforms can include for example, stationary ground stations, aircraft, satellites, ships, underwater vehicles, buoys, ground vehicles, people and animals, towers, balloons, rockets, rifles, etc. If a sensor is attached to a gimbal that rotates or moves, that gimbal is also considered a platform which may itself be attached to another platform.

While the internal characteristics may determine the extent and the resolution of an image, it is the location and orientation of the platform that determines where the observations are being measured. One typically determines the location and orientation of the sensor by determining its location and orientation on the platform (i.e. its mounting location and orientation), and then measuring the location and orientation of the platform relative to some geodetic datum.

For georeferencing purposes, it is important that one is able to accurately determine both the location and orientation (and sometimes the velocity) of the platform relative to the observation target. Thus, it is important to sensors, platforms, and observation targets as bodies with orientation and not simply geographical points which have only location. In this document, we will use the term "**position**" to include both the location and orientation of an object as determined by referencing its coordinate frame to some external coordinate frame. For platforms, the orientation of the platform is often referred to as its "**attitude**", while the combination of location, orientation, velocity , and acceleration is referred to as its "**state**".

The position of any platform can typically be expressed as a location, measured perhaps using GPS sensor, and orientation, measured perhaps using a compass or **Inertial Measurement Unit** (IMU). However, for various platform types, their position can be expressed different location models (e.g. through Keplerian orbital elements for satellites) or based on particular assumptions (e.g. the platform will be constrained to remain on the earth's surface for a ground-based vehicle).

6.2.2 Satellite Platforms

Satellite sensors are typically designed to remain within a particular orbit path about a planet. Thus, while a satellite's location can be expressed through absolute coordinates (perhaps from GPS measurements), its location can also be expressed quite accurately through the use of orbital element (parameters) based on the Kepler Laws of orbital dynamics. While the GPS measurements can improve location measurements, the orbital element model has an advantage of requiring a very small amount of data and of being able to predict satellite location in the future.

Accurate measurements of satellite orientation are more often a source of inaccuracy than its location. For orientation, satellite platforms are often designed to remaining a relatively constant

orientation relative to its velocity vector and the direction to nadir (i.e. the point directly below the platform measured perpendicular to the planet's ellipsoid). Any deviations from this orientation, either intentional or unintentional, are measured as pitch-roll-yaw values given in degrees of rotation.

6.2.3 Aircraft Platforms

The state of airborne platforms are usually measurements of 3D location given by GPS sensors, additional altitude measurements provided by altimeters, true heading measurements (angular degrees from North) provided by digital compass, pitch-roll-yaw orientation provided by an IMU sensor system, and velocity and acceleration derived from changes in the GPS position.

6.2.4 Ground-based Vehicle Platforms

Ground-based platforms are those that are confined to remaining on the surface of the planet. Thus, an automotive vehicle carrying a camera system (e.g. a police car) is constrained to follow the terrain of the ground at its current location. The location of a ground-based vehicle, as well as its velocity and acceleration, can be determined by GPS measurements. However, for a typical GPS on the Earth's surface, the accuracy of latitude and longitude measurements is better than the accuracy of altitude. The accuracy of the altitude measurement can be improved by utilizing Differential GPS (DGPS) which augments GPS using a network of ground-based reference stations. One can also utilize digital terrain models to determine the altitude of the vehicle at a particular latitude and longitude, but the accuracy of this method is constrained by the accuracy of the terrain model.

Most ground-based vehicles do not carry sensors capable of accurately determining the vehicles orientation. In such a case, one must assume that the front of the vehicle is oriented in the instantaneous velocity direction and that the pitch of the vehicle is coincident with the slope of the terrain in the direction of the velocity vector at that point. The introduction of a digital compass and inclinometers can greatly improve the accuracy of georeferencing of camera systems on ground-based vehicles.

6.2.5 Water-borne Platforms

Water-borne platforms can either be loosely constrained to the water's surface or be able to move about within the volume of water. In the former case, measurement of the platform's state is similar to that of a ground-based vehicle, with the added complication that the water vehicle is much more likely to bob up and down and to roll and pitch relative to the constraining surface.

For the case of the underwater vehicle, such as Autonomous Underwater Vehicles (AUV), the state parameters of the vehicle are similar to an aircraft although the methods of obtaining position are different. "AUVs can navigate inside a net of acoustic beacons; this is known as Long Base Line (LBL) navigation. When a surface reference such as a support ship is available, Ultra-short baseline (USBL) positioning is used to calculate where the subsea vehicle is relative to the known (GPS) position of the surface craft by means of acoustic range and bearing measurements. When it is operating completely autonomously, the AUV will surface and take its own GPS fix. Between position fixes and for precise maneuvering, an inertial navigation system onboard the AUV measures the acceleration of the vehicle and Doppler velocity

technology is used to measure rate of travel. A pressure sensor measures the vertical position. These observations are filtered to determine a final navigation solution". [Wikipedia]

6.2.6 Human Platforms

Imagery taken from human platforms is typically the most difficult to georeference. Even if the position of the human is accurately known from GPS sensors, the orientation of the human is rarely measured. The best one can typically do to georeference imagery from hand-held cameras is to match up the orientation of the image to correlate with know objects in the image.

6.2.7 Static In-situ Platforms

The position of a static platform can be accurately measured using GPS, while the orientation of a static camera system can be determined by compass and inclinometer, or by determining the direction of known objects in the field of view. Unfortunately, accurate position measurements of ground-based surveillance cameras are rarely taken or reported.

Often, position might be provided strictly relationship to the location of known objects or the union of two or more geospatial features, while orientation can be rather vague. For example, a traffic camera might be listed as observing westbound traffic at the intersection of Bradford Drive and Sparkman Drive in Huntsville, Alabama. It is anticipated that as sensor webs and the desire to provide precise geolocation from all cameras becomes ubiquitous, that more accurate measurement of location and orientation will be provided for surveillance and environmental camera systems.

7 Platform Models

While platform or sensor position is often considered as part of the sensor model, it is important to understand the position of the platform can be determine through a variety of models, depending on the accuracy desired and on the potential dynamic constraints of the system. For example, one might use different platform models for satellite, which are typically constrained to orbital dynamics, than one might use for an aircraft or vehicle constrained to the land or ocean surface. By considering the platform model separately from the sensor model, where appropriate, one can not only apply the best model for the task, but one can easily apply the same sensor model regardless of whether that sensor is mounted on a satellite, aircraft, ship, or ground station.

It is also important to understand that the parameters that feed into these platform models (e.g. location, orientation, true heading, etc. are themselves sensor measurements and not metadata in the truest sense. While this may seem like simply a discussion in semantics, how one views these data (or parameters) often determines how these data are made available to the models and is therefore highly relevant to the scope of this document.

7.1 Relationship of the sensor to a platform [Botts and Robin, 2007]

A sensor system is typically composed of one or more sensors mounted on a platform. Only the sensor element is viewed as being able to measure physical quantities. A platform such as an

aircraft (carrying a frame camera) may be able to determine its own instantaneous orientation and position, and in such a case, these measurements would be obtained by other sensors attached to the platform. In many models, platforms and sensors are treated as separate entities. Typically a sensor will be modeled as consisting of one or more detectors, whereas a platform will be modeled as a system that contains all of the sensors and defines positional and temporal relationships among them.

The platform may be stationary or moving with respect to the geodetic reference frame. For moving platforms, the geospatial position and orientation of a sensor is often derived from the platform on which it is mounted. It is therefore often beneficial to relate the sensor's coordinate frame to the coordinate frame of the platform's (e.g., through mounting angles and position) and then depend on the platform for determining geospatial positioning. It is also possible to ignore the platform's reference frame, and to strictly provide positional information relative to other sensors that provide positional observations (e.g., a GPS or IMU device). All of the frames of reference can in general be dynamic or static. Figure 7 illustrates the relationship of a sensor's frame (in pink) that is fixed but has been translated and rotated relative to the moving spacecraft frame (in black).

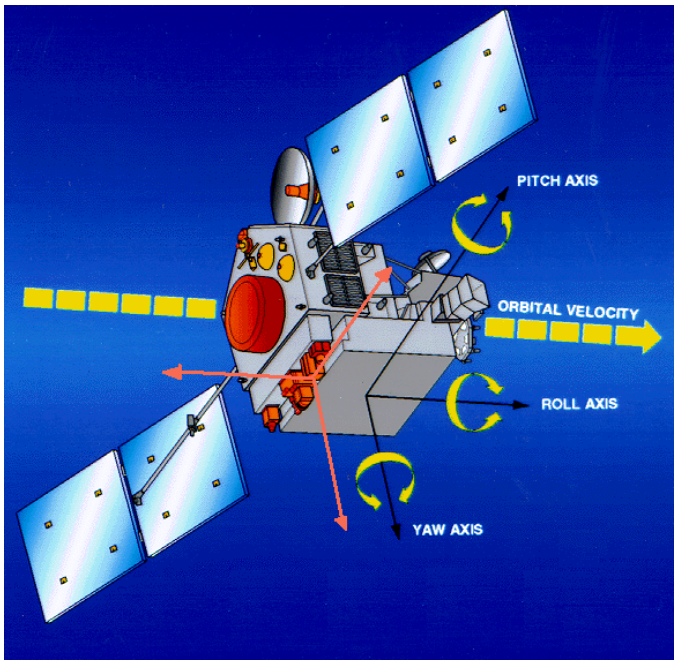

Figure 7. Relationship of sensor frame (pink) to the moving platform frame (black).

Based on the type of sensor and on the characteristics of the platform, a sensor system can be classified according to Table 1. Based on the dynamics of the platform, a sensor system may be fixed (stationary) or mobile (dynamic). Based on the sensor characteristics, a sensor system may measure either in-situ (in place) or remotely. Thus, a remote sensing atmospheric profiler might be fixed to the ground (fixed remote) or attached to an aircraft (mobile remote). Similarly, an in-

situ water quality sensor might be attached to a fixed station (fixed in-situ) or to a boat (mobile in-situ).

As previously discussed, we separate the description of the sensor from that of its platform. The main importance of the associated platform(s) is in providing the relationship of the sensor and its observations to some relevant external coordinate system (for example, a geospatial reference system).

Measures / Mobility	In-Situ	Remote
Fixed	Stationary O2 Probe	Doppler Radar station
Mobile	"Diving" Salinity probe	Airborne LIDAR

Table 1. Relationships between in-situ and remote sensors on mobile and fixed platforms.

In lieu of providing sensor location relative to a platform oriented coordinate reference system, one can also choose to simply relate sensor locations relative to other sensors that are responsible for providing location (e.g., a GPS sensor) and attitude (e.g., an inertial navigation system).

7.2 Coordinate reference systems [Botts and Robin, 2007]

All geometric and temporal characteristics of a sensor system must be related to a specified coordinate reference system (CRS). Typically, definitions for sample geometry, look angle, and collection geometry are often described relative to the sensor's CRS. In such cases, it is only through the sensor's relationship to its mount and platform(s), that the sensor and its measurements can be related to an external CRS, such as geographic latitude and longitude.

This is accomplished by defining CRSs and describing their relationships to one another. The relationship between CRSs can be accomplished either by describing a transform process between the coordinate reference systems or by defining the state of the object relative to a CRS. For instance, an individual sample's geometry (e.g., shape and size) is defined in the localized coordinates of that sample. Its relationship to a sensor's frame may be specified through a collection geometry definition. The sensor's CRS may, in turn, be related to its platform's CRS through its mounting angles and position. Finally, the platform's CRS is related to a geospatial CRS by defining its position and orientation within that CRS. The successive transformation of each of these coordinate frames into its parent CRS provides the information necessary to georegister the sensor's measurements. It is also possible, using particular sensor models, to relate the sample geometry and position directly to a geospatial CRS.

For a remote sensor, it is necessary to determine the intersection of a pixel's look ray and the surface of the sensor's target (e.g., the Earth's ellipsoid). Typically the look angle and the

sensors target are transformed into a common spatial reference frame, such as the Earth-Centered Earth-Fixed (ECEF) or Earth Centered Inertial (ECI) reference system. For in-situ sensors, the process is typically much easier.

There are also two common local Cartesian reference systems useful for georeferencing models, typically referred to as ENU (East-North-Up) and NED (North-East-Down). These systems are defined for a particular point on or near the Earth's surface. The point location serves as the origin of the reference frame, where the x, y, and z axes correspond to the local east, north, and up directions, respectively, for the ENU system, and local north, east, and down directions, respectively, for the NEDS system. The z axis is considered perpendicular to the ellipsoid at the origin's location.

The CRS concept will also be applied to temporal domain when applicable. One local time frame that is useful for defining the geometry and dynamics of scanners is seconds past the start of a scan (scan start time). Also, for some sensor systems, time is recorded relative to a local clock or the start of the mission. In such cases, time frames and their relationship to "Earth time" should be defined within the platform and sensor models.

7.3 Orbit model

An orbit model provides parameters and methods for defining the position, velocity, and acceleration of an orbiting body as a function of time. The algorithms are based on the Kepler laws for orbit dynamics, adjusted for drag between the object and the atmosphere. The parameters of the model define an elliptical orbit within an Earth-Centered-Inertial coordinate frame. One useful aspect of the orbit models that satellite orbits are regular and thus able to propagated beyond the last orbit measurement. Thus, it is possible within the predictability of the orbit, to determine exactly when a satellite-borne sensor could view a given area in the future.

For attitude determination of satellites, one either assumes that the orientation of the satellite is kept in a nadir-pointing orientation, or one expects to receive attitude adjustments measurements (e.g. pitch, roll, yaw) derived from an IMU, a star tracker system, or a combination of both.

Various parameter formats and model algorithms exist, including NASA two-line-elements (TLE) used by the Norad SGP4 propagator. An example of a SensorML-encoded SGP4 orbit model is presented in Annex D.

7.4 Explicit and Inertial positioning

With the prevalence of the GPS navigation, many sensor systems now use GPS measurements for determining location, direction, and velocity of the platform. The accuracy of the GPS can be enhanced using supplementary systems such as Differential GPS and a high-speed Inertial Navigation System (INS). A GPS unit is suitable for a wide range of platforms including satellites, aircraft, ground-based vehicles, above-surface boats, and humans.

GPS units are not capable, however, of determining the orientation of the sensor system, unless one can assume that the platform's coordinate system is always constant relative to the platform's velocity direction. This assumption might be sufficient for a ground-based vehicle for

example. Otherwise, an orientation measurement must be measured using either an IMU or a digital compass based on the magnetic field. Typically, orientation measurements are given as pitch, roll, and true heading (angular measurement between the forward axis of the platform and true North), or as pitch, roll, yaw (angular measurement between the forward axis of the platform and its velocity vector) combined with a velocity vector within a geospatial coordinate system (such as ECEF). One must be aware that the order and sign of pitch, roll, and yaw measurements is not always is not consistent between different systems; it is thus important that the encodings of these models prove an explicit definition of the CRS and the order of rotations in the system.

Inertial positioning models begin with a know location and orientation and through measured changes in orientation and velocity, determine the current position. Inertial systems are subject to accumulated errors with time and are today more often used to supplement explicit positioning where available.

8 Sensor Models

As has been discussed previously, the geolocation of remotely sensed data typically utilizes a series of models for transforming data between image domain (e.g. pixel x, y) and geospatial domain (e.g. latitude, longitude, and altitude or ECEF x,y,z). The term “Sensor Model” is sometimes used to denote the entire series of transformations and other times used to denote strictly the model describing the geometry and dynamics of the sensor itself.

There are both rigorous sensor models and polynomial fit sensor models. A rigorous sensor model is defined here as one that describes the geometry and physical dynamics of the instrument and provides the ability to utilize this information along with position and orientation of the platform in order to derive geolocation of the sensor data. Mathematical sensor models are typically derived using a rigorous model, perhaps augmented by human interaction. These general mathematical models typically hide the physical characteristics of the sensor and allow for geolocation of sensor data through the use of polynomial functions.

In essence, however, each of these models allows one to transform vectors or positions from one spatial-temporal domain to another. Models exist that allow one to derive a ground position given a particular x and y position in image space (i.e. an “image-to-ground” model), as well as allowing one to determine a position in image space given a particular position within geospatial space (i.e. a “ground-to-image” model). In both cases, the parameters required by the models tend to be the same and include the sensor’s optical or scanning characteristics (sometimes referred to as the “internal geometry”), and the location, orientation, and dynamics of the sensor’s platform (sometimes referred to as the “external geometry”).

Figure 8 illustrates a typical rigorous model for image-to-ground transformation. Given a particular row and column position in the image coordinate system, the Sensor Model provides a look vector within the Sensor CRS (i.e. within the sensor’s local reference frame). Using a Sensor mounting Model, which in essence the location and orientation of the sensor within the platform’s CRS, the look ray is transformed into the platform’s reference frame. The Platform Model utilizes known or derived state of the platform (i.e. location, orientation, and velocity) to transform the look vector into a geospatial CRS. This geospatial reference frame is typically

ECEF or ECI, since the latitude-longitude-altitude reference system is not a true Cartesian reference frame (note: if a latitude-longitude-altitude is used for projecting vector rays, it is only for very localized systems and one should account for the distortion of vector paths resulting from spherical curvature). Finally, one may apply atmospheric refraction models to account for bending of the look vector as it passes between target and sensor, at which point an ray intersection model can determine at what geospatial location the ray interacts with the ellipsoid, the terrain, or some other object within its view.

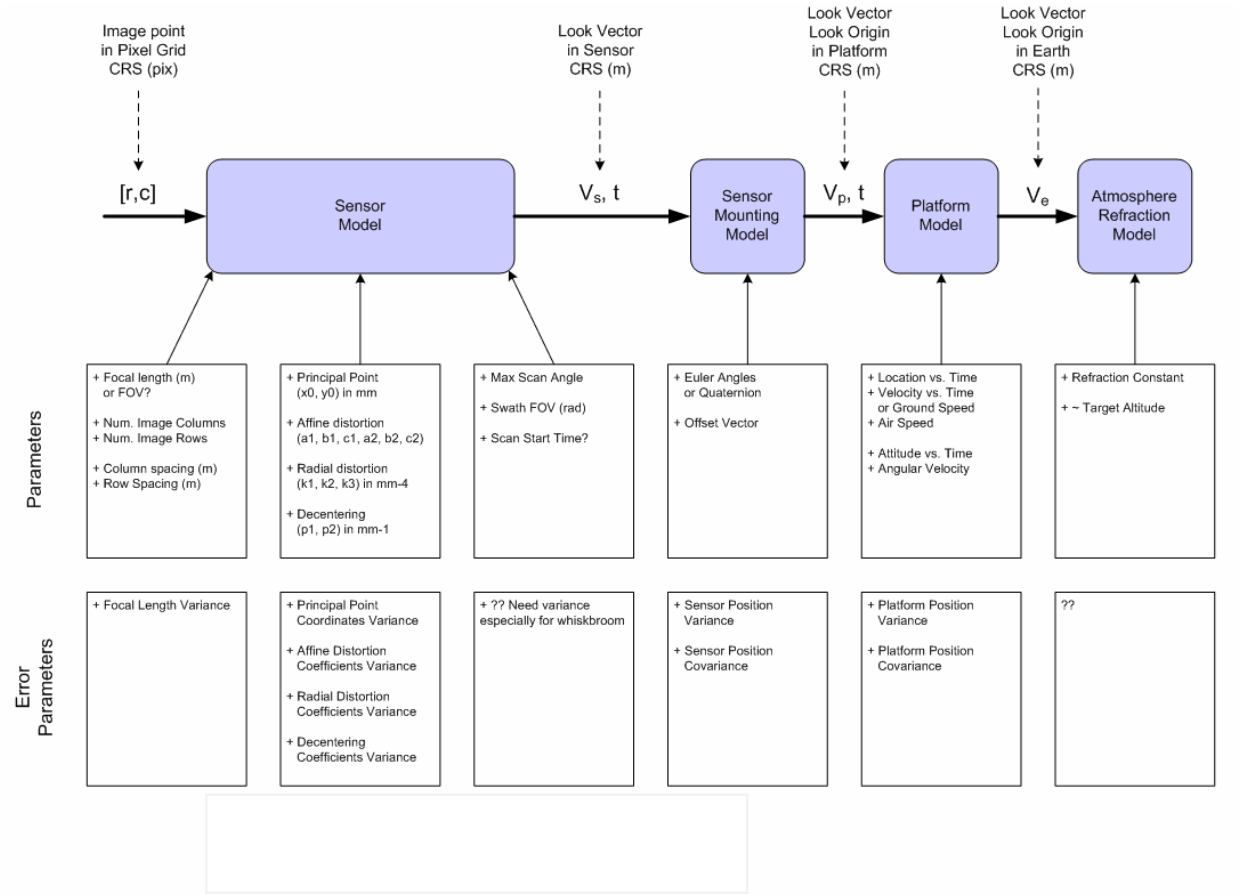

Figure 8. A modular view of a rigorous georeferencing model for image-to-ground models, showing the relationship of the "internal" sensor model to the "external" platform models [Botts, Robin, Berthiau (2007)]

There are also functional fit models that utilize polynomial relationships to allow image-to-ground or ground-to-image transformations. Although such models have hidden the parameters and information required by rigorous models, the coefficients and other parameters required for these functional fit models have typically been derived using precise rigorous models.

Standard sensor models are currently being developed by the Community Sensor Model Working Group (CSMWG) for frame sensors, pushbroom and whiskbroom scanners, profilers,

Synthetic Aperture Radar (SAR), and the RPC and RSM functional fit models. These models are expected to become a part of the developing ISO TC211 19130 standard for sensor models.

8.1 Frame Sensor

The frame sensor model is defined in detail within the Community Sensor Model (CSM) document, Frame Sensor Model Metadata Profile Supporting Precise Geolocation [CSM FSM07], which defines required and optional parameters, as well as the equations to perform ground-to-image geolocation using the collinearity equation. Those details will not be repeated within this document.

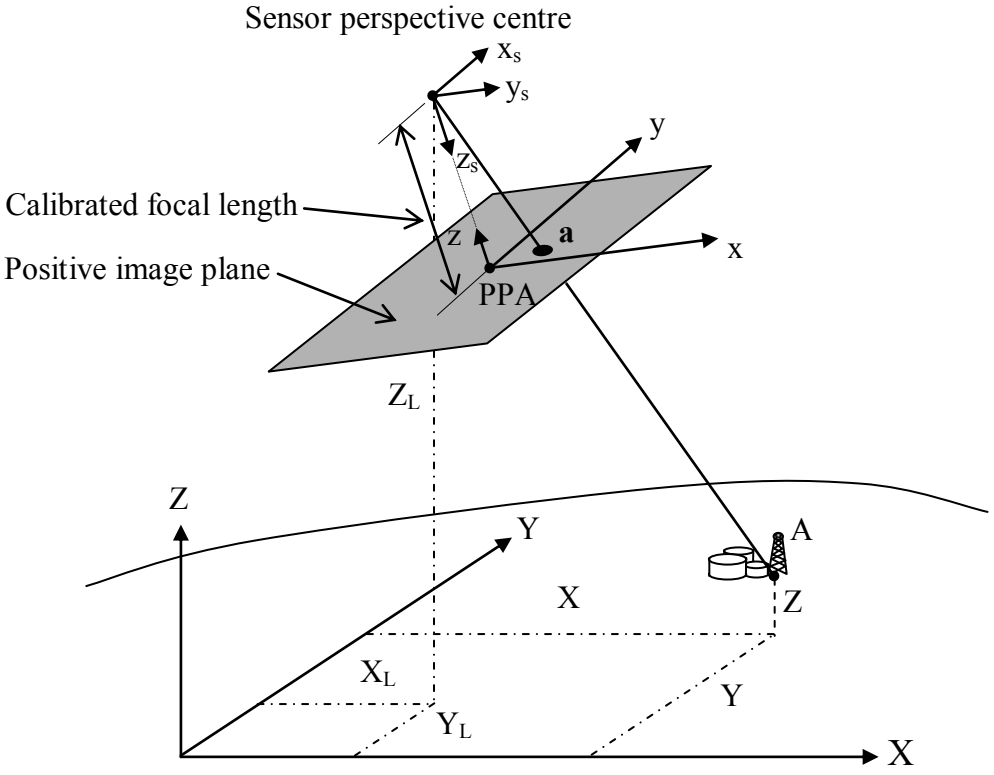

Figure 9. Illustration of a frame sensor showing the relationship between the image plane and the observe object.

The frame sensor model parameters primarily provide a description of the optical properties of the sensor system, including the calibrated focal length (see Fig. 9), a collection of known distortions within the optical system, and the dimensions of the detector array. Fig. 10 illustrates how these parameters could be applied in a modular process chain. Furthermore, Table 2 list the parameters associated with the interior orientation of the frame sensor. In addition, one would need to account for the platform and mounting model parameters to provide complete knowledge required for georeferencing an image.

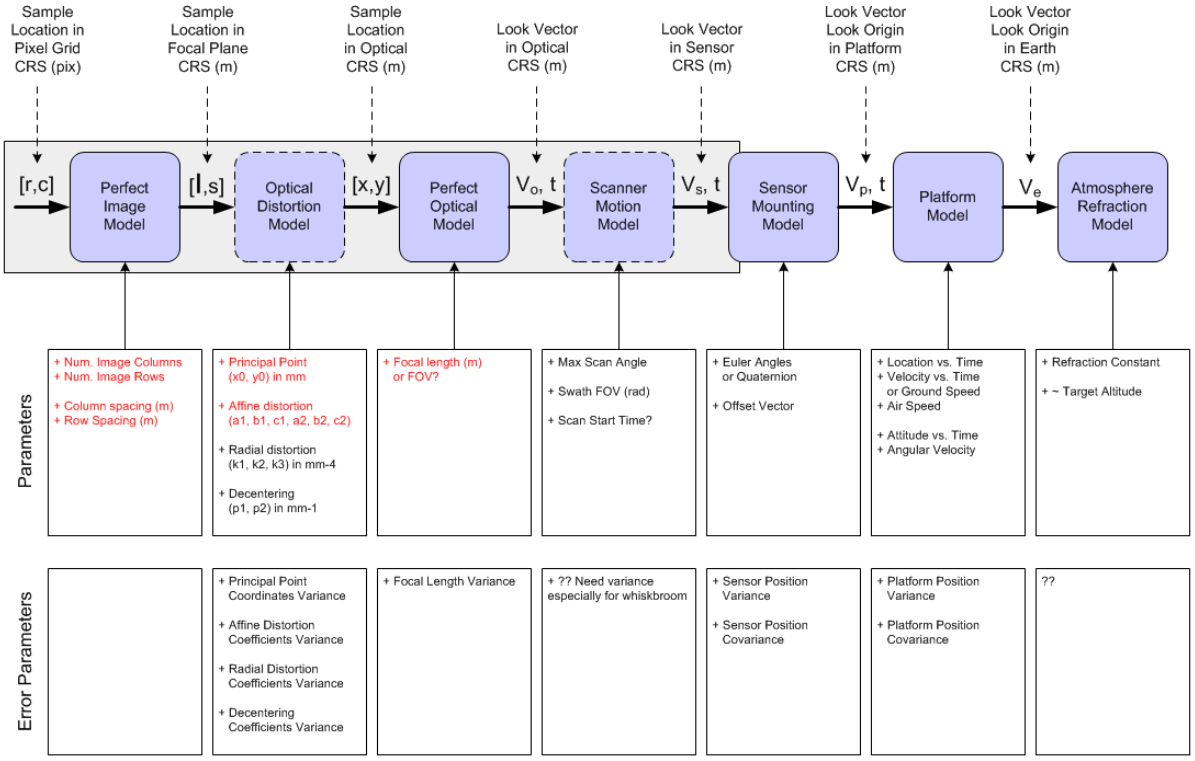

Figure 10 . A modular view of the frame sensor model, accounting for distortions to the perfect image and optical models [Botts, Robin, Berthiau (2007)]

ID	*Parameter*	*Definition*	*Field and Cross-reference to NITF, this, or other standards*
3	Number of Columns in Image	C, the number of columns in the image. (unitless)	NCOLS
4	Number of Rows in Image	R, The number of rows in the image. (unitless)	NROWS
5	Collection Start Time	The date and time at the start of the sensor activation.	COLLECTION_START_TIME
12	Sensor Calibrated Focal Length	*f*, lens calibrated focal length (mm); effective distance from optical lens to sensor element(s).	FOCAL_LENGTH
13	Principal point off-set, x-axis	x_0, x-coordinate of the foot of the perpendicular dropped from perspective center (focal point) of the camera lens onto the collection array. (mm).	PRIN_OFFSETX

ID	*Parameter*	*Definition*	*Field and Cross-reference to NITF, this, or other standards*
14	Principal point off-set, y-axis	y_0, y-coordinate of the foot of the perpendicular dropped from perspective center (focal point) of the camera lens onto the collection array. (mm).	PRIN_OFFSETY
19	Column Spacing	Column spacing measured at the center of the image; distance in the image plane between adjacent pixels within a row, measured in feet or meters.	COL_SPACING
20	Row Spacing	Row spacing measured at the center of the image; distance in the image plane between corresponding pixels of adjacent rows, measured in feet or meters.	ROW_SPACING
21	Various distortions a_1,b_1,c_1,a_2,b_2,c_2	representing 2 scales, rotation, skew, and 2 shifts	
22	Column axis offset	C_ℓ, linear translation from the image upper-left corner pixel to the focal array origin (mm), s axis	COL_AXIS_OFFSET
23	Row axis offset	C_s, linear translation from the image upper-left corner pixel to the to focal array origin (mm), ℓ axis	ROW_AXIS_OFFSET
24	Lens radial distortion coefficients	k_1 (mm^-2), k_2 (mm^-4), k_3 (mm^-6), lens radial distortion coefficients	DISTOR_RAD1 DISTOR_RAD2 DISTOR_RAD3
25	Decentering lens correction coefficients	$p_1(mm^{-1})$, $p_2(mm^{-1})$	DECEN_LENS1 DECEN_LENS2

Table 2. List of optical parameters for the Frame Sensor Model; see [CSM FSM07] for details.

The frame sensor model is appropriate for precise mapping cameras, digital cameras, video cameras, and as described below, pushbroom scanners. Figure 11 shows georeferencing of High-Definition video on-board an Unmanned Air Vehicle (UAV) using the CSM Frame Sensor Model.

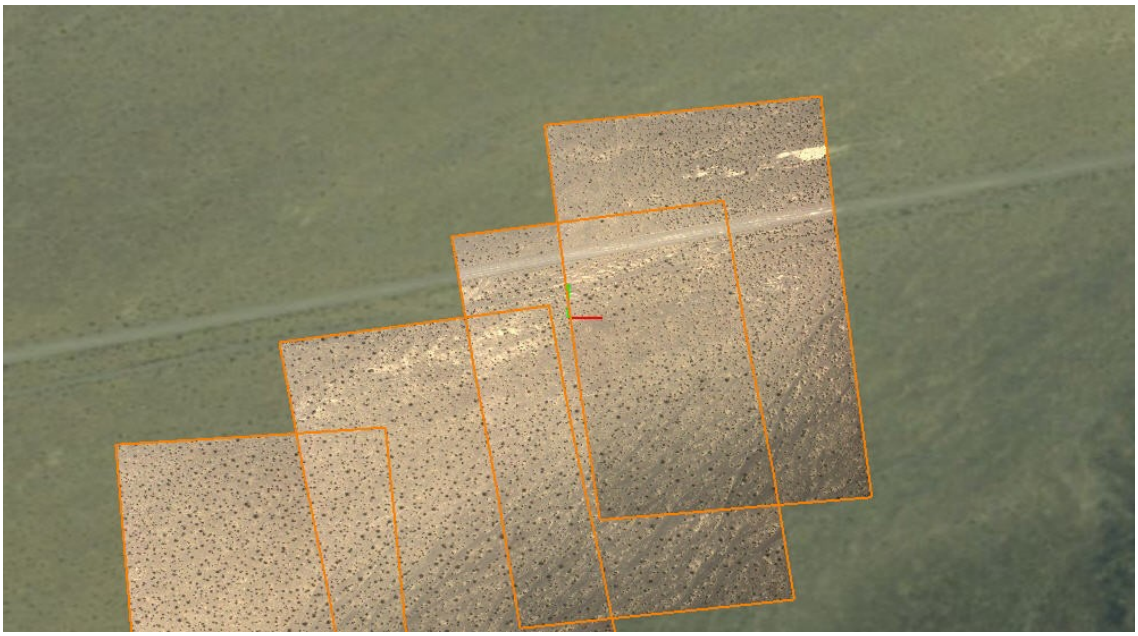

Figure 11. On-demand geolocation of the GSI KSM-HD High-Definition video imagery taken from the Tigershark UAV during Empire Challenge 08; uses the SensorML encoding of the CSM Frame Sensor model along with the UAH Open Source SensorML process execution engine software.

8.1.1 Scanners – Pushbroom and Whiskbroom

A pushbroom scanner is one in which one or more rows of pixels are sampled simultaneously. Thus, it can simply be modeled as a frame sensor where the number of scan lines is limited (possibly equal to 1) relative to the number of scan samples.

The whiskbroom scanner sensor model can also be derived from the frame sensor model, with the added complication that each pixel can be sampled at a different time. Thus the dynamics of the scanning system must be accounted for as shown in Fig. 10. However, because the scanning process is often the result of a rotating mirror or rotating platform, parameters typically associated with optical system, such as focal length and radial distortions are often not used to describe whiskbroom scanners. Instead, whiskbroom scanner models typically provide some measure or derivation of cross scan angle as a function of seconds past scan start time. These angles can be derived from scan start and stop angles, along with the rate of scan, or the angles can delta times can be explicitly stated. Figures 3 and 4 show the on-demand georeferencing of two satellite-borne Earth Observation sensors using SensorML-encodings of a simple scanner model.

The scanner models have not yet been fully defined by the Community Sensor Model Working Group or ISO19130 committee. This section will be augmented and refined in later versions of the document.

8.1.2 Profilers

A profiler takes measurements a various distances along the line of sight. Profilers can either have a fixed look direction (as seen in Fig. 6), or can be combined with a scanning action (as in Fig. 5). Thus, the profiler sensor model can be an extension of the previously described models, in which additional information is given regarding the distance and size of bins along the look vector. Like the scanner models, the profiler models have not yet been fully defined by the Community Sensor Model Working Group or ISO19130 committee. This section will be augmented and refined in later versions of the document.

8.1.3 Explicit Look Vector Model

The models described above each provide a means for calculating look vectors for any pixel position in the image array based on a set of well-defined sensor characteristics. An additional rigorous model simply provides the look vector within the sensor's reference for every pixel in a scan line or image frame. For each pixel position, this would typically consist of a pair of vector coordinates in distance or angle units. These vectors are typically either pre-calculated from the models above or they are explicitly measured on a laser bench. In either case, the distortions in the system will be accounted for. While this model reduces the need to calculate these vectors, it by far requires the largest number of parameters of all models.

8.1.4 Functional Fit models – RPC, RSM, Tie-Point

A functional fit model provides a algebraic relationship between the position of a pixel within image space and the location of that observation on the ground. While most such models utilize rigorous models in their derivation, they tend to replace and hide the specifics of the rigorous model. This can be done for the purpose of obfuscating information needed for executing rigorous models (e.g. sensor position and imaging capabilities) or for providing a simple mathematical approach to georeferencing imagery and providing error measurements.

The most common functional fit sensor models include the Rational Polynomial Coefficients (RPC) model, the Replacement Sensor Model (RSM), and the Tie-Point model. Unlike the RPC and RSM, the tie-point model is not based on a rigorous sensor model, but is often used to augment the geolocation resulting from the other models. Instead of utilizing knowledge of the sensor system, the tie-point model is based on associating pixels in the image with known object locations on the ground, and stretching and contracting parts of the image to best match up these locations, a process often referred to as "rubber sheeting".

8.1.4.1 RPC Details

The RPC model has been in use by government agencies and commercial imagery suppliers for many years. The model is fairly simple, but the error reporting and the ability to adjust parameters for greater accuracy are currently limited. The RPC model is being replaced by the RSM model which utilizes the same sets of basic parameters but adds a greater number of error parameters, as well as parameters that can be used to adjust the accuracy of the model. Still the RPC model provides an example of functional fit models. In OWS-5, the application of SensorML-encoded RPC models was demonstrated for on-demand georeferencing of image tiles streaming through a JPIP server.

The parameters required for the RPC model consist of offset and scale values for each dimension, as well as four sets of twenty coefficients. The SensorML encoding of the RPC model is provided in Appendix C.

The following description of the RPC algorithm is adapted and modified from "RPC Data File Format" by Gene Dial of Space Imaging.

Given a targetY (e.g. latitude), targetX (e.g. longitude), and targetZ (e.g. height above ellipsoid), the RPC model calculates the decimal imageX and imageY location within the image CRS.

Begin by converting latitude, longitude, and height to normalized units:

$P = (targetY - targetYOffset) / (targetYScale);$

$L = (targetX - targetXOffset) / (targetXScale);$

$H = (targetZ - targetZOffset) / (targetZOffset);$

Let:

$XN_1, XN_2, XN_3, \ldots, XN_{20}$ = the xNumeratorCoefficients 1 through 20

$XD_1, XD_2, XD_3, \ldots, XD_{20}$ = the xDenominatorCoefficients 1 through 20

$XN_1, XN_2, XN_3, \ldots, XN_{20}$ = the XNumeratorCoefficients 1 through 20

$YD_1, YD_2, YD_3, \ldots, YD_{20}$ = the yDenominatorCoefficients 1 through 20

The scaled X and Y coordinates are calculated:

$Y = XNum / yDen;$

$X = xNum / xDen;$

where

$$XNum = XN_1 + (XN_2 \times L) + (XN_3 \times P) + (XN_4 \times H) + (XN_5 \times L \times P) + (XN_6 \times L \times H) + (XN_7 \times P \times H) + (XN_8 \times L^2) + (XN_9 \times P^2) + (XN_{10} \times H^2) + (XN_{11} \times P \times L \times H) + (XN_{12} \times L^3) + (XN_{13} \times L \times P^2) + (XN_{14} \times L \times H^2) + (XN_{15} \times L^2 \times P) + (XN_{16} \times P^3) + (XN_{17} \times P \times H^2) + (XN_{18} \times L^2 \times H) + (XN_{19} \times P^2 \times H) + (XN_{20} \times H^3)$$

$$yDen = YD_1 + (YD_2 \times L) + (YD_3 \times P) + (YD_4 \times H) + (YD_5 \times L \times P) + (YD_6 \times L \times H) + (YD_7 \times P \times H) + (YD_8 \times L^2) + (YD_9 \times P^2) + (YD_{10} \times H^2) + (YD_{11} \times P \times L \times H) + (YD_{12} \times L^3) + (YD_{13} \times L \times P^2) + (YD_{14} \times L \times H^2) + (YD_{15} \times L^2 \times P) + (YD_{16} \times P^3) + (YD_{17} \times P \times H^2) + (YD_{18} \times L^2 \times H) + (YD_{19} \times P^2 \times H) + (YD_{20} \times H^3)$$

$$yNum = YN_1 + (YN_2 \times L) + (YN_3 \times P) + (YN_4 \times H) + (YN_5 \times L \times P) + (YN_6 \times L \times H) + (YN_7 \times P \times H) + (YN_8 \times L^2) + (YN_9 \times P^2) + (YN_{10} \times H^2) + (YN_{11} \times P \times L \times H) + (YN_{12} \times L^3) + (YN_{13} \times L \times P^2) + (YN_{14} \times L \times H^2) + (YN_{15} \times L^2 \times P) + (YN_{16} \times P^3) + (YN_{17} \times P \times H^2) + (YN_{18} \times L^2 \times H) + (YN_{19} \times P^2 \times H) + (YN_{20} \times H^3)$$

$$yDen = YD_1 + (YD_2 \times L) + (YD_3 \times P) + (YD_4 \times H) + (YD_5 \times L \times P) + (YD_6 \times L \times H) + (YD_7 \times P \times H) + (YD_8 \times L^2) + (YD_9 \times P^2) + (YD_{10} \times H^2) + (YD_{11} \times P \times L \times H) + (YD_{12} \times L^3) + (YD_{13} \times L \times P^2) + (YD_{14}$$

$$x\ L\ x\ H^2) + (YD_{15}\ x\ L^2\ x\ P) + (YD_{16}\ x\ P^3) + (YD_{17}\ x\ P\ x\ H^2) + (YD_{18}\ x\ L^2\ x\ H) + (YD_{19}\ x\ P^2\ x\ H) + (YD_{20}\ x\ H^3)$$

De-normalize X and Y:

$$imageY = Y * imageYScale + imageYOffset$$

$$imageX = X * imageXScale + imageXOffset$$

8.1.4.2 Replacement Sensor Model (RSM) Overview

The RSM model is consider an improvement on the RPC model. Like the RPC model, it utilizes a ratio of polynomial equations for determining the image x and y locations, but it has additional capabilities for reporting error and for providing adjustments to the model after the rigorous model to RSM model transformation has occurred. Thus model will be covered in more detail in later versions of this document.

9 Encoding Sensor Models

Within the OGC standards body, there are currently three means of possibly encoding sensor model parameters, including (1) Sensor Model Language, or SensorML, (2) Image Geopositioning Metadata (IGM), and (3) TransducerML (TML) standards. As will be shown in the sections below, the differences in the packaging and content of each encoding result from the initial assumptions and base classes from which they are derived. For example, SensorML treats sensor models as a specialization of a general process model, whereas IGM derives sensor models as an image operation derived from a general coordinate transform. TML starts with a streaming data model that can be related to a sensor model through a Transducer Characteristic Frame.

9.1 Sensor Model Language (SensorML) [OGC 07-000]

SensorML, Version 1.0.1, is a general XML framework for describing processes surrounding the observations. These include the process of measurement itself (i.e. the description of the sensor as a process), and the processes that can be applied to the observations (e.g. geolocation, image processing, or digital signal processing). Thus, in SensorML, sensor models are encoded as a specialization of a process, where inputs, outputs, parameters, and method of the process is explicitly defined.

SensorML enables robust definitions of **sensor models** for providing geolocation of observations from remote sensors. SensorML supports both rigorous geolocation models and mathematical geolocation models. Different mathematical models can be designed to define a sample location within a variety of coordinate systems, including the local sensor frame, the local frame for the associated platform, or a geographic coordinate reference frame.

To support complex processing or to allow modularization and reuse of multiple process models (e.g. sensor models, platform models, spatial transform models), processes can be linked within process chains or physical systems. The components and the data connections between the inputs, outputs, and parameters of these processes are explicitly defined in SensorML. If each ProcessModel in the chain can be related to an associated class or module in software, then a SensorML process chain becomes a protocol-agnostic process that can be executed on-demand within various execution engines. The execution environments could include the existing UAH Open Source SensorML execution engine or ultimately the NGA Image Processing Library, various commercial image processing and analysis programs, or a BPEL workflow engine.

Dynamic values required by the sensor models, such as platform location and attitude or changes to sensor scan characteristics can be provided inline as time-tagged swe:DataArray, or they can be fed into a SensorML process chain from an SOS request or through a real-time stream.

SensorML can provide support in two ways. One is by providing information about the platform's state, the mounting angles, and sensor parameters (e.g. optical properties) within various component descriptions within the System. It is then up to specialized software to pull out this information and apply it to georeference observations coming from that sensor. The second approach is to provide a SensorML process chain or model that explicitly describes how to utilize this information to generate the imagery geolocation on demand. This approach does not require software specifically designed for georeferencing imagery, but instead can utilize a SensorML-enabled processing engine to execute any SensorML-described process on demand. Furthermore, this process chain can be referenced within the PositionList within the sml:System description.

Currently, as part of the CSMWG efforts, SensorML profiles are being created for the CSM sensor models being developed and proposed as international standards. Because of the fine granularity of the schema desired, these SensorML profiles are being developed in RelaxNG. Examples of SensorML-encoded sensor models are provided in Appendices B (Frame Sensor), C (RPC Model), and E (Orbital Model).

9.2 Image Geopositioning Metadata (IGM) [OGC 07-031]

The IGM schema is a GML 3.2 Application Schema for image geopositioning metadata, which is also an Application Schema of ISO 19139. This image geopositioning metadata is designed for use by a separately specified Image Geopositioning Service (IGS) that adjusts the georeferencing coordinate transformations of images. This XML schema encodes image georeferencing coordinate transformations with associated parameter error statistics. These georeferencing coordinate transformations can use many possible image geometry (or sensor) models that can be encoded using extensions of this Application Schema.

This Application Schema also encodes object point positions measured in one of more images and optional object coordinates, with associated position error statistics. These object points can be tie points, control points, and check points. Error statistics are represented as variance-covariance matrices, representing both absolute and relative accuracies. These covariance matrices are used to represent correlations between the accuracies of different parameters, coordinates, and positions.

In the IGM Package, sensor models are supported using the ObjectImageTransformation which is a specialization of the coordinate transform classes in GML. Thus, the input and output of the transformation is expected to be an image, which will be transformed between the source and target CRSes which are explicitly defined.

An example of an IGM-encoded sensor model is provided in Appendix D.

9.3 TransducerML (TML) [OGC 06-010r6]

TML is a set of models for defining the response characteristics of a transducer and schema for transporting multiplexed data aggregates through an XML-based stream. TML takes a data-centric approach to detectors and actuators and defines various time-tagged data structures referred to as Transducer Characteristic Frames (TCF). Since the TCF is simply a data structure, it can include observations as well as parameters such as those from a sensor model. Georeferencing of imagery or other sensor observations from a TML data stream requires specialized software that understands how to georeference observations from specific sensor types. Alternatively, one can link appropriate components of a TML data stream into a SensorML-encoded process chain for geolocation. This approach was demonstrated using IR scanner data from the NASA AIRDAS UAV.

9.4 Summary of the Sensor Model encodings in OGC

Although each of these standards has each taken a slightly different approach, they are not incompatible with one another. Although this has not yet been demonstrated, the information shared between these approaches is similar, such that translation from one encoding to another is believed to be achievable. The differences in the packaging and content result from the initial assumptions and base classes from which the sensor model classes are derived.

In essence, SensorML derives the various sensor models from the general sml:ProcessModel used to support any type of non-physical process. As with all processes defined in SensorML, inputs and outputs of these processes are explicitly defined, while the sensor model parameters are supported by the parameters element. If the process is a transformation from one reference frame to another, the source and target CRSes are provided as attributes within the input and output Vectors. Chaining of these processes is supported using the sml:ProcessChain class, in which the components and connections between inputs, outputs, and parameters are explicitly defined. The process algorithm is defined in the method element. Dynamic parameters can be provided as inline data arrays, or fed directly into the process chain from Sensor Observation Services.

The IGM standard supports sensor models using the igm:ObjectImageTransformation class which is a specialization of a coordinate transformation. Thus, the input and output are implied to be an image that is transformed from the source CRS to the target CRS. The current examples in IGM show sensor, mounting, and platform models combined within a single ObjectImageTransformation instance. In IGM, dynamic parameters (e.g. platform position) are typically defined as GML geometry objects (e.g. gml:Curve) and provided inline or referenced through xlink:href.

TML takes a data stream approach to sensor models an provides the ability to encode explicit look angles or sensor model parameters within its TCF data structure. Parameters are thus considered to be a part of the data stream provided from the sensor.

All of these approaches have validity. While each could function independent of one another, there may be some merit to investigating means of harmonizing these approaches in the future.

10 Delivering Sensor Model Parameters

The previous sections have briefly described three possible OGC standards for encoding Sensor Models. This section will briefly explore how these Sensor Models can be delivered to applications and services which can use them for georeferencing an imagery product.

10.1 Sensor Observation Services

The Sensor Observation Service (SOS) is a part of the OGC Sensor Web Enablement (SWE) suite of standards. Its role is to provide low-level sensor observations as well as information about the sensor system that produced the Observations. Observations are returned using the Observations and Measurements (O&M) schema, or alternately by pointing to a TML server. Information about the sensor system is returned as SensorML or TML.

In the O&M Observation specification, the sensor model information can be provided by two means. One approach is to provide the process chain or sensor model parameters within the SensorML System description that is returned using the describeSensor method for SOS. The other is to reference the SensorML geolocation process chain within the location property of the Feature of Interest for the Observation.

As discussed previously, TML provides sensor model parameters as part of the sensor's data stream.

The SWE community takes the approach that a sensor system can return a wide range of time-tagged observations in addition to those returned by the sensor itself. These might include for instance, the platform's GPS location and IMU attitude measurements, gimbal rotation measurements, status values for various components within the system, tasking commands being acted on by the system, error determinations within the system, and dynamic parameter values within the sensor models. It is vital that such measurements be treated as observations in their own right, and not as simply metadata to support the imagery sensor observations. Such measurements can be utilized completely independent of the actual imagery, for example, to discovery when a sensor looked at areas of interest or to issue alerts based on sensor conditions or view conditions. For example, in Figure 12, an SOS request retrieved only aircraft location and attitude information, which combined the CSM Frame Sensor model, provided the ability to determine where the sensor was looking at any particular time before requesting the high-volume video stream.

Figure 12. Footprints calculated using a SensorML-encoded CSM Frame Sensor Model, based on retrieving the platform's navigation data (i.e. location and attitude) from an SOS. Rather than treating such data as metadata for the imagery, treating sensor model parameters and other measurements as observations in their own right, enables increased capabilities for discovery and alerts.

10.2 Web Coverage Services

The Web Coverage Service supports the subsetting and delivery of both rectified and georeferenceable imagery. Some of the available formats, such as JPEG2000 (J2K), support the ability to encode sensor models within the image format itself. OWS 5.1 demonstrated the ability to encode sensor models in J2K by extending the GML_J2K specification to allow SensorML-encoded sensor models.

Other formats do not support sensor models within the format, and thus require that this information be pointed to separately. In WCS, this is supporting within the SpatialDomain section of the CoverageDescription. This XML description is returned in response to a describeCoverage request to the WCS. Within the current version being submitted for review and acceptance, the sensor model can either be provided as a gml:CoordinateOperation (e.g. IGM) or as a reference to a SensorML process chain or model.

10.3 JPIP

The ability to support interactively browsing of large imagery (up to 30GB) through a JPIP server was tested and demonstrated in OGC OWS-5.1. The imagery consisted of unrectified high-resolution imagery from SPOT Image and Global Images. During this test bed, it was demonstrated that a SensorML-encoded Sensor Model (in this case an RPC model) could be passed through the back channel of the JPIP stream and utilized for on-demand georeferencing.

11 Workflow Options

Regardless of the encoding or delivery of the Sensor Model using OGC services and encodings, it is possible to utilize the sensor models within a variety of workflow environments. These are briefly illustrated here using a SensorML process chain as an example. Figure 13 illustrates the application of sensor models for on-demand processing of georeferenceable imagery within a client. In this example, observations including imagery and platform navigation measurements (e.g. location and attitude) are retrieved from an SOS and fed into a SensorML process chain for georectification, styling, and display within the interactive client. An example of this process is shown in Figure 14.

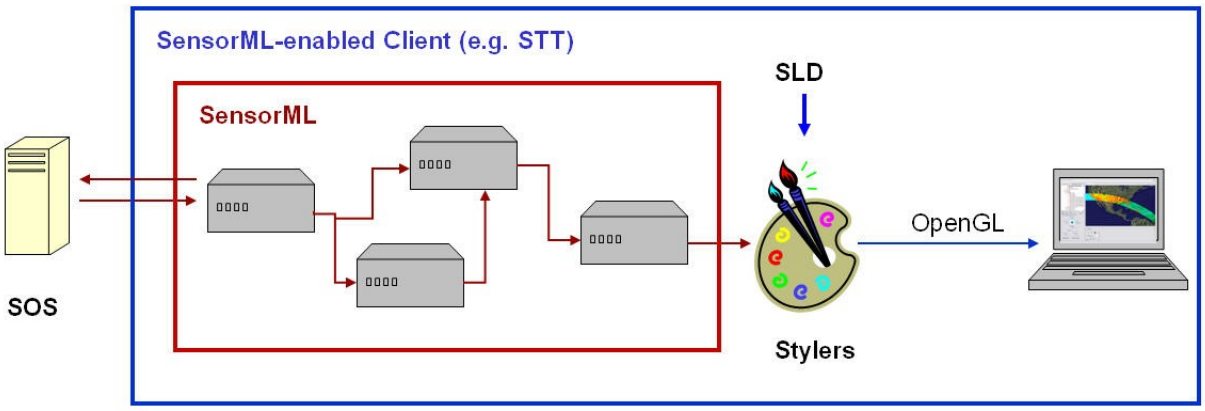

Figure 13. Illustration showing the georeferencing process residing within a client.

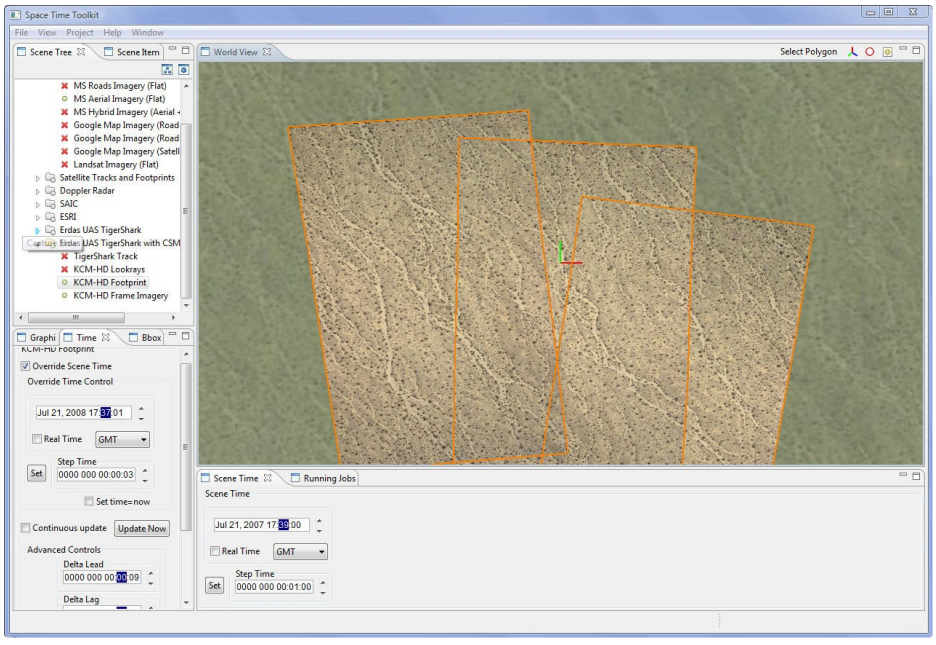

Figure 14. Example of on-demand georectification of video imagery using SensorML within the UAH Space Time Toolkit client.

Figure 15 illustrates that georectification can occur within a service, thereby allowing on-demand processing of georeferenceable imagery as a result of a web service request.

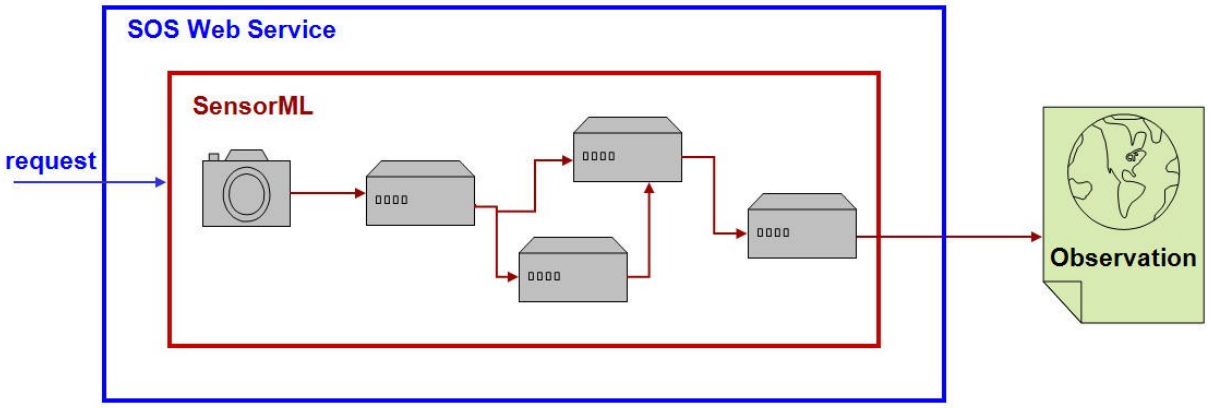

Figure 15. Illustration showing the georeferencing process residing within a service where in this case, rectified observations are provided on-demand.

Figure 16 illustrates a similar situation where the georeferencing process is utilized within a web service for on-demand processing. In this case, however, the georeferencing process allows one to determine and set appropriate gimbal and camera setting to view a particular area of interest.

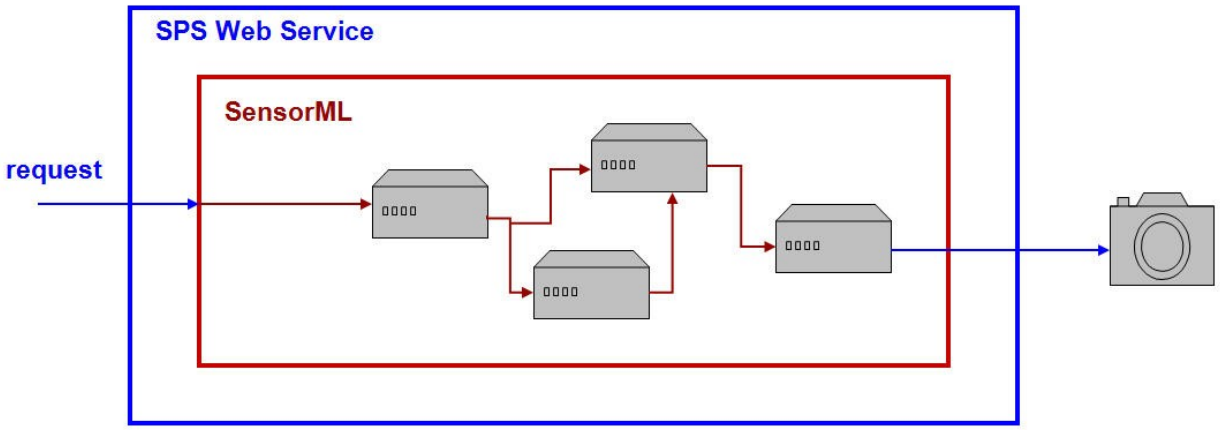

Figure 16. Illustration showing the use of a georeferencing process residing within a sensor tasking service (SPS) where in this case, the ability to georeference a sensor allows one to control the pointing of a camera to view a desired area..

Figure 17 illustrates a slight modification from the previous workflows where in this case the georeferencing process again occurs in a web service, but this service includes a styling component that creates graphical data within a standard format, such as KML, Collada, or

Virtual Earth. Figure 18 shows examples of observations delivered to the Google Earth client through the workflow illustrated in Figure 17.

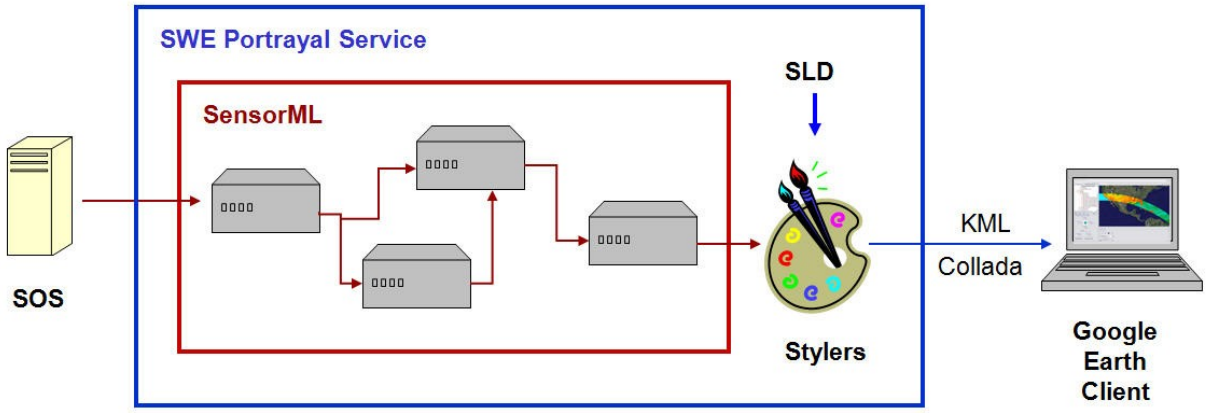

Figure 17. Illustration showing the georeferencing process residing within a service where in this case, rectified observations are provided on-demand and the resulting imagery is styled into a graphic standard (e.g. KML, Collada, or Virtual Earth) for delivery to Google Earth or Virtual Earth clients.

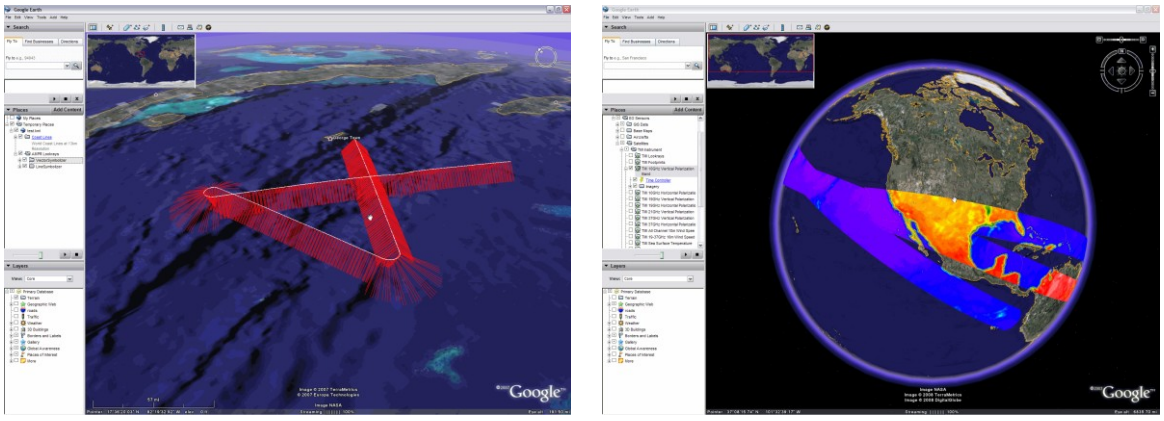

Figure 18. Examples of SOS delivered observations, georeferenced and delivered to the Google Earth client.

12 Issues

It has been demonstrated within various venues and test beds that the OGC web services and encodings are capable of robustly and efficiently supporting the geolocating and processing georeferenceable imagery. It is important that concrete examples and tutorials be made available

to the general public. This document is only a beginning to address that need. The high-level nature of presentation in this document has provided only an overview of the current state of supporting georeferenceable imagery using OGC encodings and services. A concerted effort among OGC members and editors of the various specifications is needed to provide a less confusing set of information to potential users and developers, and to provide a clear path for support georeferenceable imagery using OGC standards.

Annex A

Discussion on the nature of observations

This is a synopsis of a discussion taken from email on the nature of observations and coverages [M. Botts, 2007]. It explores the temporal and spatial aspects of sensor observations, including georeferenceable imagery, and is thus relevant to the current document. There are several levels at which one can consider what is an observation, a coverage, or a geographical representation of observations. This also explores the possible relationships between OGC web services for serving sensor data and georeferenceable imagery, including Sensor Observation Service (SOS), Web Coverage Service (WCS), and Web Feature Service (WFS).

Level 1: Pure Observation - Only time coverage is considered.

There is the concept that observations are really, in their purest form, just time-dependent measurements, regardless of the dynamics or lack of dynamics of the sensor system, and regardless of the sampling pattern. In this concept, geometries are not yet considered, but might be implied or derivable from understanding the sensor system characteristics (e.g. dynamics, sampling pattern, sampling rate, etc). There are at this point, perhaps, several different geometries that might be considered relevant depending on what one wants to do with the observations.

This is the approach that many of us have typically tried to take within the SWE effort. Observations are strictly time-based measurements, SensorML can describe the system such that various geometries (including geospatial) can be derived, and an SOS should only allow subsetting on time and property axes. Any spatial subsetting in SOS should only involve determining which sensors might be relevant (particularly for in-situ stations) but subsetting an actual coverage in SOS should be beyond scope of the SOS. That's why I typically refer to SWE services as "low-level".

Level 2: Various other coverages are considered.

In this concept, various coverage domains could be defined as suitable for a particular set of Observations. Obviously, geospatial coverage is one of those (e.g. points, lines, grids, polygons, and images), but there could also be other possible coverage domains such spectral/frequency, species, size, etc. These possible coverages might fall out of or be implied by the nature of the sampling, the dynamics of the system, or the characteristics of the sensor. But typically, there is a decision, and possibly a set of calculations, that we make to put the observations into a specific coverage domain.

This is where taking time-tagged GPS pure observations and deciding to represent it as a line geometry happens. This is where one derives a sensor’s footprint or a georectified image based on the dynamics and scanning characteristics of a remote sensor.

Although I am fully aware that WFS and WCS perhaps can and have supported Level 1 observations, I believe that it is in the Level 2 concept where these services have traditional excelled, whereas the SWE encodings and services have specifically been geared toward Level 1 concepts.

Level 2 observations might involve geometries but are still not necessarily graphical representations.

Level 3: Graphical Representation / Portrayal.

OGC has been wise in appreciating, during the design of GML, WFS, and WCS, that is important to keep data content standards independent of graphic content standards. Level 3 gets into the portrayal of an observation (i.e. the graphical view). In other words, one might take that aircraft path, make it a line defined by these points, color the line blue, and make it 3 pixels wide. Or better yet, one might make the color of the line at any point depend on the concentration of ozone measured along the path, use a particular color map. Or one could take a temperature observation and display it in big yellow letters at this location.

This is the level that VGL, KML, and Collada are best suited for. It is important to understand that this is a very important level, but that at this point we are no longer interacting with the actual observations but a graphical representation of the data. This is also where technologies like OGC's Style Layer Description (SLD) language play an important role because they provide us a way to describe the mapping from Level 1 or 2 to Level 3.

The important message of this long rant is that all levels are important, but that it is VERY important that when we design standards or semantics or when we converse on specific data types or services, that we understand at which of these levels we are dealing or desiring to deal.

Additional discussion in follow-up email:

(1) first one needs to recognize that what is represented in the output of a sensor or process in SensorML, as well as what is packaged in an om:Observation is data content, not graphics content. In other words, we may give you a time series of temperature, pressure, wind chill, etc., but we don't tell you how to portray it. One might wish to show the temperature as a time series plot or take the latest value and display it as a text label at the appropriate location.

Similarly, looking at data from a remote sensing scanner on a polar orbiting satellite, we also don't tend to tell you in the SensorML and O&M that you should portray this as a gridded image. It again is just a time series of observations where the scanning geometry characteristics of the sensor might suggest that you could show this as an image with X being along scan and Y being along satellite track, but you could also chose to display the spectral curve at a single pixel point.

The same can the true of the time-series location of an aircraft or satellite. It's a collection of time-based measurements that COULD be recognized as a line. However, unlike GML which tends to portray much of its geospatially oriented data as geometries, we have tended to focus on the idea that these are measurements in SensorML, O&M, and SOS. That is one of the reasons

why I perhaps consider most GML Features as higher-level data that can perhaps be derived from lower-level sensor Observations. The same argument can be applied to SOS versus WFS.

(2) That being said, Alex Robin and I have constantly argued both within ourselves and between ourselves, as to the importance of defining data in SWE Common as being of a particular geometry. We typically end up shying away from it, still convincing ourselves that this is a portrayal issue perhaps.

(3) That doesn't mean that we shouldn't define records and arrays as being of a particular type. This appears to be what is being done in the CSML Feature Types that you send in your email. So using the "definition" attribute in SWE Common, one might specify a DataRecord or DataArray as a type with expected components:

```
<swe:DataArray definition="urn:ogc:def:property::PointMeasurementSeries">
      <swe:elementCount>
            <swe:Count>
                  <swe:value> 120 </swe:Count>
             </swe:Count>
      </swe:ElementCount>
      <swe:elementType>
            <swe:DataRecord definition="urn:ogc:def:property::pointMeasurement">
                  <swe:field name="time">
                        <swe:Time definition="urn:ogc:def:property:observationTime"/>
                  </swe:field>
                  <swe:field name="temperature">
                        <swe:Quantitydefinition="urn:ogc:def::property::atmosphericTemperature">
                              <swe:uom code="Cel"/>
                        </swe:Quantity>
                   </swe:field>
                   <swe:field name="pressure">
                  ... etc
```

Notice that the definition attribute for the DataRecord defines it of being of type "pointMeasurement" which is helpful, but the type "pointMeasurement" would not necessarily be an XML Schema that tells you that the components must be temperature, pressure, etc.. Instead a definition or profile for "pointMeasurement" might only tell you that it has a collection of measurements and that the first component must be of type "observationTime".

Similarly, notice that the definition of the DataArray is of time "pointMeasurementSeries" which might be define as a sequential collection of time-based point measurements ("pointMeasurement").

I think these definitions for aggregate types can be very helpful in developing consistency in presenting certain types of data, and also in portrayal. I think the work done in CSML and other efforts would be VERY useful in this regard.

Annex B

Example of Frame Camera Model encoded in SensorML

Example of an CSM-based internal Frame Camera Model encoded in SensorML. In this example, all distortion parameters are supported, but some can be optional. Note that the parameters are referenced in the definitions according to CSM namespace. Also note that it is possible to provide quality indications ("standard error" in this case) for any parameters for which these values are known.

This example shows that values for the parameters can be provided inline within the appropriate XML tags, in lieu of the example in Annex C which shows how the parameters can be described first and then provided as an in-line data block or externally through a data stream.

```
<?xml version="1.0" encoding="UTF-8"?>
<?oxygen RNGSchema="frame-sensor-model.rng" type="xml"?>
<sml:ProcessModel xmlns:sml="http://www.opengis.net/sensorML/1.0" xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:gml="http://www.opengis.net/gml" xmlns:ism="urn:us:gov:ic:ism:v2" xmlns:xlink="http://www.w3.org/1999/xlink">
    <gml:description>ERDAS KCM39_60mm_prod-012</gml:description>
    <gml:name>KCM39_60mm_prod-012</gml:name>
    <sml:validTime>
        <gml:TimeInstant>
            <gml:name>Calibration Date</gml:name>
            <gml:timePosition>2007-09-19T18:20:43</gml:timePosition>
        </gml:TimeInstant>
    </sml:validTime>
    <sml:securityConstraint>
        <sml:Security ism:classification="U"/>
    </sml:securityConstraint>
    <sml:inputs>
        <sml:InputList>
            <sml:input name="Pixel Grid Coordinates">
                <swe:Vector referenceFrame="urn:ogc:def:crs:CSM:pixelGridCRS">
                    <swe:coordinate name="r">
                        <swe:Quantity>
                            <swe:uom xlink:href="urn:ogc:def:unit:CSM:pixel"/>
                        </swe:Quantity>
                    </swe:coordinate>
                    <swe:coordinate name="c">
                        <swe:Quantity>
                            <swe:uom xlink:href="urn:ogc:def:unit:CSM:pixel"/>
                        </swe:Quantity>
                    </swe:coordinate>
                </swe:Vector>
            </sml:input>
        </sml:InputList>
    </sml:inputs>
    <sml:outputs>
        <sml:OutputList>
            <sml:output name="View Vector">
                <swe:Vector referenceFrame="urn:ogc:def:crs:CSM:sensorCRS">
                    <swe:coordinate name="x">
                        <swe:Quantity/>
                    </swe:coordinate>
                    <swe:coordinate name="y">
                        <swe:Quantity/>
                    </swe:coordinate>
                    <swe:coordinate name="z">
                        <swe:Quantity/>
```

```
                </swe:coordinate>
            </swe:Vector>
        </sml:output>
    </sml:OutputList>
</sml:outputs>
<sml:parameters>
    <sml:ParameterList>
        <sml:parameter name="Focal Length">
            <swe:Quantity definition="urn:ogc:def:property:CSM:FOCAL_LENGTH">
                <swe:quality>
                    <swe:Quantity definition="urn:ogc:def:property:OGC:stdError">
                        <swe:value>5.073e-003</swe:value>
                    </swe:Quantity>
                </swe:quality>
                <swe:uom code="mm"/>
                <swe:value>60.1634</swe:value>
            </swe:Quantity>
        </sml:parameter>
        <sml:parameter name="Pixel Grid Characteristics">
            <swe:DataRecord>
                <swe:field name="Number of Rows in Image">
                    <swe:Count definition="urn:ogc:def:property:CSM:NROWS">
                        <swe:value>5389</swe:value>
                    </swe:Count>
                </swe:field>
                <swe:field name="Number of Columns in Image">
                    <swe:Count definition="urn:ogc:def:property:CSM:NCOLS">
                        <swe:value>7162</swe:value>
                    </swe:Count>
                </swe:field>
                <swe:field name="Row Spacing">
                    <swe:Quantity definition="urn:ogc:def:property:CSM:ROW_SPACING">
                        <swe:uom code="mm"/>
                        <swe:value>0.0068</swe:value>
                    </swe:Quantity>
                </swe:field>
                <swe:field name="Column Spacing">
                    <swe:Quantity definition="urn:ogc:def:property:CSM:COL_SPACING">
                        <swe:uom code="mm"/>
                        <swe:value>0.0068</swe:value>
                    </swe:Quantity>
                </swe:field>
            </swe:DataRecord>
        </sml:parameter>
        <sml:parameter name="Principal Point Coordinates">
            <swe:Vector referenceFrame="urn:ogc:def:crs:CSM:imagePlaneCRS">
                <swe:coordinate name="x0">
                    <swe:Quantity definition="urn:ogc:def:property:CSM:PRIN_OFFSETX">
                        <swe:quality>
                            <swe:Quantity definition="urn:ogc:def:property:OGC:stdError">
                                <swe:value>1.062e-003</swe:value>
                            </swe:Quantity>
                        </swe:quality>
                        <swe:uom code="mm"/>
                        <swe:value>0.3440</swe:value>
                    </swe:Quantity>
                </swe:coordinate>
                <swe:coordinate name="y0">
                    <swe:Quantity definition="urn:ogc:def:property:CSM:PRIN_OFFSETY">
                        <swe:quality>
                            <swe:Quantity definition="urn:ogc:def:property:OGC:stdError">
                                <swe:value>9.487e-004</swe:value>
                            </swe:Quantity>
                        </swe:quality>
                        <swe:uom code="mm"/>
                        <swe:value>-0.2206</swe:value>
                    </swe:Quantity>
```

```
            </swe:coordinate>
        </swe:Vector>
    </sml:parameter>
    <sml:parameter name="Affine Distortion Coefficients">
        <swe:DataRecord>
            <swe:field name="a1">
                <swe:Quantity definition="urn:ogc:def:property:CSM:DISTOR_A1">
                    <swe:value>0</swe:value>
                </swe:Quantity>
            </swe:field>
            <swe:field name="b1">
                <swe:Quantity definition="urn:ogc:def:property:CSM:DISTOR_B1">
                    <swe:quality>
                        <swe:Quantity definition="urn:ogc:def:property:OGC:stdError">
                            <swe:value>2.289e-016</swe:value>
                        </swe:Quantity>
                    </swe:quality>
                    <swe:value>2.55044e-024</swe:value>
                </swe:Quantity>
            </swe:field>
            <swe:field name="c1">
                <swe:Quantity definition="urn:ogc:def:property:CSM:DISTOR_C1">
                    <swe:value>0</swe:value>
                </swe:Quantity>
            </swe:field>
            <swe:field name="a2">
                <swe:Quantity definition="urn:ogc:def:property:CSM:DISTOR_A2">
                    <swe:value>0</swe:value>
                </swe:Quantity>
            </swe:field>
            <swe:field name="b2">
                <swe:Quantity definition="urn:ogc:def:property:CSM:DISTOR_B2">
                    <swe:quality>
                        <swe:Quantity definition="urn:ogc:def:property:OGC:stdError">
                            <swe:value>2.289e-016</swe:value>
                        </swe:Quantity>
                    </swe:quality>
                    <swe:value>2.17308e-025</swe:value>
                </swe:Quantity>
            </swe:field>
            <swe:field name="c2">
                <swe:Quantity definition="urn:ogc:def:property:CSM:DISTOR_C2">
                    <swe:value>0</swe:value>
                </swe:Quantity>
            </swe:field>
        </swe:DataRecord>
    </sml:parameter>
    <sml:parameter name="Radial Distortion Coefficients">
        <swe:DataRecord>
            <swe:field name="k1">
                <swe:Quantity definition="urn:ogc:def:property:CSM:DISTOR_RAD1">
                    <swe:quality>
                        <swe:Quantity definition="urn:ogc:def:property:OGC:stdError">
                            <swe:value>1.088e007</swe:value>
                        </swe:Quantity>
                    </swe:quality>
                    <swe:uom code="mm-2"/>
                    <swe:value>1.92709e-005</swe:value>
                </swe:Quantity>
            </swe:field>
            <swe:field name="k2">
                <swe:Quantity definition="urn:ogc:def:property:CSM:DISTOR_RAD2">
                    <swe:quality>
                        <swe:Quantity definition="urn:ogc:def:property:OGC:stdError">
                            <swe:value>2.813e-010</swe:value>
                        </swe:Quantity>
                    </swe:quality>
```

```
                        <swe:uom code="mm-2"/>
                        <swe:value>-5.14206e-010</swe:value>
                    </swe:Quantity>
                </swe:field>
                <swe:field name="k3">
                    <swe:Quantity definition="urn:ogc:def:property:CSM:DISTOR_RAD3">
                        <swe:quality>
                            <swe:Quantity definition="urn:ogc:def:property:OGC:stdError">
                                <swe:value>2.185e-013</swe:value>
                            </swe:Quantity>
                        </swe:quality>
                        <swe:uom code="mm-2"/>
                        <swe:value>-3.33356e-012</swe:value>
                    </swe:Quantity>
                </swe:field>
            </swe:DataRecord>
        </sml:parameter>
        <sml:parameter name="Decentering Coefficients">
            <swe:DataRecord>
                <swe:field name="p1">
                    <swe:Quantity definition="urn:ogc:def:property:CSM:DECEN_LENS1">
                        <swe:quality>
                            <swe:Quantity definition="urn:ogc:def:property:OGC:stdError">
                                <swe:value>2.289e-016</swe:value>
                            </swe:Quantity>
                        </swe:quality>
                        <swe:uom code="mm-1"/>
                        <swe:value>-1.25151e-024</swe:value>
                    </swe:Quantity>
                </swe:field>
                <swe:field name="p2">
                    <swe:Quantity definition="urn:ogc:def:property:CSM:DECEN_LENS2">
                        <swe:quality>
                            <swe:Quantity definition="urn:ogc:def:property:OGC:stdError">
                                <swe:value>2.289e-016</swe:value>
                            </swe:Quantity>
                        </swe:quality>
                        <swe:uom code="mm-1"/>
                        <swe:value>4.10310e-024</swe:value>
                    </swe:Quantity>
                </swe:field>
            </swe:DataRecord>
        </sml:parameter>
    </sml:ParameterList>
  </sml:parameters>
  <sml:method xlink:href="urn:ogc:def:process:CSM:frameSensorModel"/>
</sml:ProcessModel>
```

Annex C

Example of RPC Model encoded in SensorML

The following example is for an RPC model for a Global Images data set. This implementation could support an array of parameter values, such as would be needed to support different parameters for different regions of the image. However, this example only shows a parameter array with only one set of parameter values that are used for the entire image.

Using this approach, it is also possible to define the RPC model once and then to stream the appropriate sets of parameter values to support different image regions or different images. This SensorML-encoded process is able to be executable using any class that can assign the parameter values, ingest the appropriate inputs, and provide the corresponding outputs, following the algorithm defined in the method: urn:ogc:def:process:CSM:RPC:1.0".

NOTE: During the design of this profile, there was some discussion about making the encoding more compact back grouping blocks of coefficients as a simple data block. However, because of confusion in the past regarding the order of coefficients and the possibility of zero value coefficients, it was recommended by many that the model be more explicit in defining each coefficient value, even if it makes the model more verbose.

This example shows how the parameters can be described first and then provided as an in-line data block or externally through a data stream, in lieu of the example in Annex C which shows that values for the parameters can be provided inline within the appropriate XML tags.

```
<sml:SensorML xmlns:sml="http://www.opengis.net/sensorML/1.0" xmlns:ism="urn:us:gov:ic:ism:v2"
xmlns:swe="http://www.opengis.net/swe/1.0" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sensorML/1.0 http://schemas.opengis.net/sensorML/1.0.0/sensorML.xsd"
version="1.0">
    <sml:member xlink:role="urn:ogc:def:process:OGC:sensormodel">
        <sml:ProcessModel gml:id="GlobalImages_05MAY14083758-M1BS-005693793010_RPC">
            <gml:description>Global Images M1BS (05MAY14083758-M1BS-005693793010) –
                Rational Polynomial Coefficients</gml:description>
            <sml:identification>
                <sml:IdentifierList>
                    <sml:identifier>
                        <sml:Term definition="urn:ogc:def:identifier:OGC:sensorID">
                            <sml:value>GlobalImages-M1BS</sml:value>
                        </sml:Term>
                    </sml:identifier>
                </sml:IdentifierList>
            </sml:identification>
            <sml:classification>
                <sml:ClassifierList>
                    <sml:classifier>
                        <sml:Term definition="urn:ogc:def:classifier:OGC:processType">
                            <sml:value>urn:ogc:def:process:CSM:RPC:1.0</sml:value>
                        </sml:Term>
                    </sml:classifier>
                </sml:ClassifierList>
            </sml:classification>
            <sml:securityConstraint>
                <sml:Security ism:classification="U"/>
```

```
</sml:securityConstraint>
<sml:inputs>
    <sml:InputList>
        <sml:input name="target_location">
            <swe:Vector definition="urn:ogc:def:data:OGC:locationVector"
                referenceFrame="urn:ogc:def:crs:EPSG:6.0:4329">
                <swe:coordinate name="x">
                    <swe:Quantity definition="urn:ogc:def: property:OGC:angle" axisID="X">
                        <gml:name>longitude</gml:name>
                        <swe:uom code="deg"/>
                    </swe:Quantity>
                </swe:coordinate>
                <swe:coordinate name="y">
                    <swe:Quantity definition="urn:ogc:def: property:OGC:angle" axisID="Y">
                        <gml:name>latitude</gml:name>
                        <swe:uom code="deg"/>
                    </swe:Quantity>
                </swe:coordinate>
                <swe:coordinate name="z">
                    <swe:Quantity definition="urn:ogc:def: property:OGC:distance" axisID="Z">
                        <gml:name>altitude</gml:name>
                        <swe:uom code="m"/>
                    </swe:Quantity>
                </swe:coordinate>
            </swe:Vector>
        </sml:input>
    </sml:InputList>
</sml:inputs>
<sml:outputs>
    <sml:OutputList>
        <sml:output name="image_location">
            <swe:Vector definition="urn:ogc:def:property:OGC:locationVector"
                referenceFrame="urn:ogc:def:crs:OGC:ImageCRSpixelCenter:RSA_SCENE1">
                <swe:coordinate name="x">
                    <swe:Quantity definition="urn:ogc:def: property:OGC:distance" axisID="X">
                        <swe:uom xlink:href="urn:ogc:def:unit:OGC:pixel"/>
                    </swe:Quantity>
                </swe:coordinate>
                <swe:coordinate name="y">
                    <swe:Quantity definition="urn:ogc:def: property:OGC:distance" axisID="Y">
                        <swe:uom xlink:href="urn:ogc:def:unit:OGC:pixel"/>
                    </swe:Quantity>
                </swe:coordinate>
            </swe:Vector>
        </sml:output>
    </sml:OutputList>
</sml:outputs>
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
<!--RPC Parameters-->
<!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
<sml:parameters>
    <sml:ParameterList>
        <sml:parameter name="rpc_parameter_series">
            <swe:DataArray>
                <swe:elementCount>
                    <swe:Count>
                        <swe:value>1</swe:value>
                    </swe:Count>
                </swe:elementCount>
                <swe:elementType name="rpc_parameter_set">
                    <swe:DataRecord definition="urn:ogc:def:data:CSM:rpcParameters">
                        <!-- -->
                        <swe:field name="image_region">
                            <swe:DataRecord>
                                <swe:field name="zone_minX">
                                    <swe:Quantity>
                                        <swe:uom xlink:href="urn:ogc:def:unit:OGC:pixel"/>
```

```
                    </swe:Quantity>
                </swe:field>
                <swe:field name="zone_minY">
                    <swe:Quantity>
                        <swe:uom xlink:href="urn:ogc:def:unit:OGC:pixel"/>
                    </swe:Quantity>
                </swe:field>
                <swe:field name="zone_maxX">
                    <swe:Quantity>
                        <swe:uom xlink:href="urn:ogc:def:unit:OGC:pixel"/>
                    </swe:Quantity>
                </swe:field>
                <swe:field name="zone_maxY">
                    <swe:Quantity>
                        <swe:uom xlink:href="urn:ogc:def:unit:OGC:pixel"/>
                    </swe:Quantity>
                </swe:field>
            </swe:DataRecord>
        </swe:field>
        <!-- -->
        <swe:field name="image_adjustment">
            <swe:DataRecord>
                <swe:field name="image_x_offset">
                    <swe:Quantity>
                        <swe:uom xlink:href="urn:ogc:def:unit:OGC:pixel"/>
                    </swe:Quantity>
                </swe:field>
                <swe:field name="image_x_scale">
                    <swe:Quantity/>
                </swe:field>
                <swe:field name="image_y_offset">
                    <swe:Quantity>
                        <swe:uom xlink:href="urn:ogc:def:unit:OGC:pixel"/>
                    </swe:Quantity>
                </swe:field>
                <swe:field name="image_y_scale">
                    <swe:Quantity/>
                </swe:field>
            </swe:DataRecord>
        </swe:field>
        <!-- -->
        <swe:field name="target_adjustment">
            <swe:DataRecord>
                <swe:field name="target_x_offset">
                    <swe:Quantity>
                        <swe:uom code="deg"/>
                    </swe:Quantity>
                </swe:field>
                <swe:field name="target_x_scale">
                    <swe:Quantity/>
                </swe:field>
                <swe:field name="target_y_offset">
                    <swe:Quantity>
                        <swe:uom code="deg"/>
                    </swe:Quantity>
                </swe:field>
                <swe:field name="target_y_scale">
                    <swe:Quantity/>
                </swe:field>
                <swe:field name="target_z_offset">
                    <swe:Quantity>
                        <swe:uom code="m"/>
                    </swe:Quantity>
                </swe:field>
                <swe:field name="target_z_scale">
                    <swe:Quantity/>
                </swe:field>
```

```
        </swe:DataRecord>
    </swe:field>
    <!-- -->
    <swe:field name="x_numerator_coefficients">
        <swe:DataRecord gml:id="PolyCoeff"
            definition="urn:ogc:def: property:CSM:rpcCoefficients">
            <swe:field name="constant">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="x">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="y">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="z">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="xx">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="xy">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="xz">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="yy">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="yz">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="zz">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="xxx">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="xxy">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="xxz">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="xyy">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="xyz">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="xzz">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="yyy">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="yyz">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="yzz">
                <swe:Quantity/>
            </swe:field>
            <swe:field name="zzz">
                <swe:Quantity/>
            </swe:field>
        </swe:DataRecord>
```

```
                                    </swe:field>
                                    <!-- other three coefficient record descriptions same as above -->
                                    <swe:field name="x_denominator_coefficients" xlink:href="#PolyCoeff"/>
                                    <swe:field name="y_numerator_coefficients" xlink:href="#PolyCoeff"/>
                                    <swe:field name="y_denominator_coefficients" xlink:href="#PolyCoeff"/>
                                    <!-- -->
                                    <swe:field name="error_parameters">
                                        <swe:DataRecord>
                                            <swe:field name="error_bias">
                                                <swe:Quantity>
                                                    <swe:uom xlink:href="urn:ogc:def:unit:OGC:pixel"/>
                                                </swe:Quantity>
                                            </swe:field>
                                            <swe:field name="error_random">
                                                <swe:Quantity>
                                                    <swe:uom xlink:href="urn:ogc:def:unit:OGC:pixel"/>
                                                </swe:Quantity>
                                            </swe:field>
                                        </swe:DataRecord>
                                    </swe:field>
                                </swe:DataRecord>
                            </swe:elementType>
                            <swe:encoding>
                                <swe:TextBlock decimalSeparator="." tokenSeparator="," blockSeparator=" "/>
                            </swe:encoding>
                            <swe:values>
                        0,0,7168,67584,3436,3476,33380,33407,2.722750000000000e+01,6.476000000000000e-01,
                        -.373990000000000e+01,6.128000000000000e-01,1206,500,1.413196000000000e-01,
                        5.456349000000000e+00,-5.443061000000000e+00,-9.216955000000000e-04,
                        3.565091000000000e-02,5.332187000000000e-03,5.838429000000000e-03,-1.079730000000000e-01,
                        -9.729955999999999e-02,9.760980000000001e-06,1.644718000000000e-04,5.002155000000000e-02,
                        -9.859079000000000e-03,-2.407938000000000e-04,1.867470000000000e-02,-4.335796000000000e-02,
                        2.417365000000000e-04,3.560836000000000e-03,-3.104646000000000e-03,-2.498503000000000e-06,
                        1.000000000000000e+00,-2.522379000000000e-02,-1.391303000000000e-03,-1.225341000000000e-03,
                        3.010631000000000e-02,7.525694000000001e-04,6.370131000000000e-04,2.880643000000000e-02,
                        8.030175000000001e-03,-4.472419000000000e-05,-7.383849000000000e-05,-3.706672000000000e-05,
                        -1.439143000000000e-04,-1.172269000000000e-06,2.300775000000000e-03,5.944449000000001e-04,
                        6.306701000000000e-07,-1.578395000000000e-05,-8.359896999999999e-06,1.877496000000000e-08,
                        1.708999000000000e-02,-5.396680000000000e-01,-5.681499000000000e-01,-1.189343000000000e-03,
                        -1.277246000000000e-02,1.240736000000000e-04,1.293826000000000e-04,-2.786896000000000e-03,
                        -4.745238000000000e-03,-1.236625000000000e-05,2.330231000000000e-05,-4.737473000000000e-04,
                        1.979220000000000e-03,4.130015000000000e-04,1.544374000000000e-03,1.681885000000000e-03,
                        4.320199000000000e-04,6.604090000000000e-06,1.384148000000000e-05,6.331177000000000e-07,
                        1.000000000000000e+00,3.584233000000000e-04,1.827953000000000e-05,1.647259000000000e-04,
                        1.313396000000000e-03,6.429174000000000e-06,9.747711000000000e-06,-1.589882000000000e-04,
                        1.177451000000000e-03,-7.649890000000000e-04,-1.643904000000000e-05,8.232680000000000e-06,
                        4.517580000000000e-04,3.976897000000000e-06,2.824726000000000e-04,3.834750000000000e-04,
                        7.641012000000000e-06,-2.343787000000000e-06,-4.520179000000000e-06,3.503771000000000e-07,
                        2.144000000000000e+01,3.000000000000000e+00
                            </swe:values>
                        </swe:DataArray>
                    </sml:parameter>
                </sml:ParameterList>
            </sml:parameters>
            <sml:method xlink:href="urn:ogc:def:process:CSM:RPC:1.0"/>
        </sml:ProcessModel>
    </sml:member>
</sml:SensorML>
```

Annex D

Example of a Sensor Model encoded in the Image Geopositioning Metadata standard (IGM) [OGC 07-031]

The sensor model in IGM is encapsulated within the ObjectImageTransformation, which references or includes sensor and image parameters. Below is an example of the ObjectImage Transformation and referenced SensorParameterValues.

--- ObjectImageTransformation ---

```
<igm:ObjectImageTransformation gml:id="Transformation999.999">
    <gml:identifier codeSpace="IGM">Transformation999.999</gml:identifier>
    <gml:scope>Domain of targetCRS</gml:scope>
    <gml:operationVersion>0.0.0</gml:operationVersion>
    <gml:sourceCRS xlink:href="urn:ogc:def:crs:EPSG:6.8:4979"/>
    <gml:targetCRS xlink:href="urn:ogc:def:crs:OGC:0.0:ImageCRSpixelCenter:TBDimageID"/>
    <gml:method xlink:href="templateFrameOperationMethod1.xml"/>
    <!-- ============================================== -->
    <igm:parameterValue>
        <ImageParameterValues>
            <igm:image xlink:href="imageInfoFile999.xml#ImageInfo999"/> <!-- Reference to
ImageInfo element for this image -->
            <igm:inGroup xlink:href="adjustedGroupFile999.xml#IAdjustedGroup999"/> <!--
Reference to AdjustedGroup element for this adjusted set of image parameter values  -->
            <igm:imageParametersStatus
codeSpace="../imageParametersStatusValues.xml">adjusted</igm:imageParametersStatus>
            <igm:sensorParameterValues
xlink:href="sensorParametersFile999.999.xml#SensorParameters999.999"/> <!-- Reference to
SensorParameterValues element for this image -->
            <igm:footprint>
                <gml:Polygon srsName="urn:ogc:crs:EPSG:6.0:9999" gml:id="Polygon999">
                    <gml:exterior>
                        <gml:LinearRing>
                            <gml:pos>999 999</gml:pos>
                            <gml:pos>999 999</gml:pos>
                            <gml:pos>999 999</gml:pos>
                            <gml:pos>999 999</gml:pos>
                        </gml:LinearRing>
                    </gml:exterior>
                    <gml:interior> <!-- Repeated for each void in exterior ring -->
                        <gml:LinearRing>
                            <gml:pos>999 999</gml:pos>
                            <gml:pos>999 999</gml:pos>
                            <gml:pos>999 999</gml:pos>
                            <gml:pos>999 999</gml:pos>
                        </gml:LinearRing>
                    </gml:interior>
                </gml:Polygon>
            </igm:footprint>
            <igm:adjustableParameters
xlink:href="adsjustableParametersFile999.xml#AdjustableParameters999"/> <!-- Reference to
AdjustableParameters element for this image -->
            <!-- ============================================== -->
            <igm:imageAccuracySummary>
                <igm:ImageAccuracySummary>
                    <igm:CE uom="m">99.9</igm:CE>
                    <igm:LE uom="m">99.9</igm:LE> <!-- Omit when not stereoscopic image -->
                    <igm:horizontalShear uom="m">99.9</igm:horizontalShear> <!-- Omit when
not applicable -->
```

```
                    <igm:verticalShear uom="m">99.9</igm:verticalShear> <!-- Omit when not
stereoscopic image -->
                </igm:ImageAccuracySummary>
            </igm:imageAccuracySummary>
            <!-- ============================================== -->
            <igm:sensorPosition>
                <gml:Point gml:id="Point999" srsName="LSR">
                    <gml:pos>999 999 999</gml:pos>
                </gml:Point>
                <igm:operationParameter
xlink:href="frameOperationMethodFile.xml#SensorPosition"/> <!-- Reference to correct
OperationParameter element -->
            </igm:sensorPosition>
            <!-- ============================================== -->
            <igm:sensorAttitude>
                <gml:Vector gml:id="Angles999" srsName="AnglesInLSR">
                    <gml:pos>9.99 9.99 9.99</gml:pos>
                </gml:Vector>
                <igm:operationParameter
xlink:href="frameOperationMethodFile.xml#SensorAttitude"/> <!-- Reference to correct
OperationParameter element -->
            </igm:sensorAttitude>
            <!-- ================================================ -->
            <igm:group xlink:href="frameOperationMethodFile.xml#ImageOrientation"/> <!--
Reference to correct OperationParameterGroup element -->
        </ImageParameterValues>
    </igm:parameterValue>
</igm:ObjectImageTransformation>
```

--- Sensor Parameters ---

```
<SensorParameterValues gml:id="SensorParameters999.999">
    <gml:identifier codeSpace="IGM">SensorParameters999.999</gml:identifier>
    <igm:sensor> <!-- Association to ImageSensor element for image -->
        <igm:ImageSensor gml:id="ImageSensor999">
            <gml:description>TBD</gml:description>
            <gml:identifier codeSpace="IGM">ImageSensor999</gml:identifier>
        </igm:ImageSensor>
    </igm:sensor>
    <igm:sensorParameterStatus
codeSpace="../imageParametersStatusValues.xml">initial</igm:sensorParameterStatus>
    <!-- ============================================== -->
    <focalLength>
        <FocalLength>
            <gml:value uom="mm">999</gml:value>
            <igm:operationParameter xlink:href="frameImageMethodFile.xml#FocalLength"/> <!-
- Reference to correct OperationParameter element  -->
        </FocalLength>
    </focalLength>
    <!-- ============================================== -->
    <ppOffsetPosition>
        <PPoffset>
            <gml:valueList uom="mm">0.999 0.999</gml:valueList>
            <igm:operationParameter xlink:href="frameImageMethodFile.xml#PPoffset"/> <!--
Reference to correct OperationParameter element  -->
        </PPoffset>
    </ppOffsetPosition>
    <!-- ============================================== -->
    <igm:group xlink:href="frameImageMethodFile.xml#OpticalPerspective"/> <!-- Reference to
correct OperationParameterGroup element  -->
</SensorParameterValues>
```

Annex E

Example of a Keplerian Orbit Model (SGP4) encoded in SensorML

```
<?xml version="1.0" encoding="UTF-8"?>
<sml:SensorML xmlns:sml="http://www.opengis.net/sensorML/1.0" xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ism="urn:us:gov:ic:ism:v2"
xsi:schemaLocation="http://www.opengis.net/sensorML/1.0 http://schemas.opengis.net/sensorML/1.0.0/sensorML.xsd"
version="1.0">
    <sml:member xlink:role="urn:ogc:def:role:OGC:locationModel">
        <sml:ProcessModel gml:id="SGP4_OrbitalModel_PROCESS">
            <!-- METADATA SECTION -->
            <gml:description>NORAD Orbital Elements for SPOT 4 </gml:description>
            <!-- Identification and Classification -->
            <sml:identification>
                <sml:IdentifierList>
                    <sml:identifier>
                        <sml:Term definition="satelliteName">
                            <sml:value>SPOT 4</sml:value>
                        </sml:Term>
                    </sml:identifier>
                    <sml:identifier>
                        <sml:Term definition="satelliteNumber">
                            <sml:value>25260</sml:value>
                        </sml:Term>
                    </sml:identifier>
                    <sml:identifier>
                        <sml:Term definition="internationalDesignator">
                            <sml:value>98017A</sml:value>
                        </sml:Term>
                    </sml:identifier>
                </sml:IdentifierList>
            </sml:identification>
            <sml:classification>
                <sml:ClassifierList>
                    <sml:classifier>
                        <sml:Term definition="ephemerisType">
                            <sml:value>SGP4</sml:value>
                        </sml:Term>
                    </sml:classifier>
                </sml:ClassifierList>
            </sml:classification>
            <!-- Time Constraints - valid for about 1-1/2 days -->
            <sml:validTime>
                <gml:TimePeriod>
                    <gml:beginPosition>2006-02-21T08:00:00Z</gml:beginPosition>
                    <gml:endPosition>2006-02-22T20:00:00Z</gml:endPosition>
                </gml:TimePeriod>
            </sml:validTime>
            <!-- Security Constraints - unclassified -->
            <sml:securityConstraint>
                <sml:Security ism:classification="U"/>
            </sml:securityConstraint>
            <!-- INPUTS DEFINITION -->
            <sml:inputs>
                <sml:InputList>
                    <sml:input name="time">
                        <swe:Time definition="urn:x-ogc:def:phenomenon:time"
                            referenceFrame="urn:ogc:def:crs:julianTime">
                            <swe:uom code="s"/>
                        </swe:Time>
```

```
            </sml:input>
        </sml:InputList>
    </sml:inputs>
    <!-- OUTPUTS DEFINITION -->
    <sml:outputs>
        <sml:OutputList>
            <sml:output name="ECI_position">
                <swe:Vector referenceFrame="urn:ogc:def:crs:eci_wgs84">
                    <swe:coordinate name="x">
                        <swe:Quantity>
                            <swe:uom code="m"/>
                        </swe:Quantity>
                    </swe:coordinate>
                    <swe:coordinate name="y">
                        <swe:Quantity>
                            <swe:uom code="m"/>
                        </swe:Quantity>
                    </swe:coordinate>
                    <swe:coordinate name="z">
                        <swe:Quantity>
                            <swe:uom code="m"/>
                        </swe:Quantity>
                    </swe:coordinate>
                </swe:Vector>
            </sml:output>
            <sml:output name="ECI_velocity">
                <swe:Vector referenceFrame="urn:ogc:def:crs:eci_wgs84">
                    <swe:coordinate name="x">
                        <swe:Quantity>
                            <swe:uom code="m/s"/>
                        </swe:Quantity>
                    </swe:coordinate>
                    <swe:coordinate name="y">
                        <swe:Quantity>
                            <swe:uom code="m/s"/>
                        </swe:Quantity>
                    </swe:coordinate>
                    <swe:coordinate name="z">
                        <swe:Quantity>
                            <swe:uom code="m/s"/>
                        </swe:Quantity>
                    </swe:coordinate>
                </swe:Vector>
            </sml:output>
        </sml:OutputList>
    </sml:outputs>
    <!-- PARAMETERS DEFINITION -->
    <sml:parameters>
        <sml:ParameterList>
            <sml:parameter name="elements">
                <swe:DataRecord definition="urn:ogc:def: property:OGC::noradElements">
                    <swe:field name="epochYear">
                        <swe:Count definition="urn:ogc:def: property:OGC::gregorianYear">
                            <swe:value>2006</swe:value>
                        </swe:Count>
                    </swe:field>
                    <swe:field name="epochDay">
                        <swe:Quantity>
                            <swe:uom xlink:href="urn:ogc:def:unit:decimalDOY"/>
                            <swe:value>52.33269901</swe:value>
                        </swe:Quantity>
                    </swe:field>
                    <swe:field name="bstar">
                        <swe:Quantity>
                            <swe:value>0.11897E-4</swe:value>
                        </swe:Quantity>
                    </swe:field>
```

```
                                <swe:field name="inclination">
                                    <swe:Quantity>
                                        <swe:uom code="deg"/>
                                        <swe:value>98.7187</swe:value>
                                    </swe:Quantity>
                                </swe:field>
                                <swe:field name="rightAscension">
                                    <swe:Quantity>
                                        <swe:uom code="deg"/>
                                        <swe:value>128.3968</swe:value>
                                    </swe:Quantity>
                                </swe:field>
                                <swe:field name="eccentricity">
                                    <swe:Quantity>
                                        <swe:value>.0000952</swe:value>
                                    </swe:Quantity>
                                </swe:field>
                                <swe:field name="argOfPerigee">
                                    <swe:Quantity>
                                        <swe:uom code="deg"/>
                                        <swe:value>101.8476</swe:value>
                                    </swe:Quantity>
                                </swe:field>
                                <swe:field name="meanAnomaly">
                                    <swe:Quantity>
                                        <swe:uom code="deg"/>
                                        <swe:value>258.2808</swe:value>
                                    </swe:Quantity>
                                </swe:field>
                                <swe:field name="meanMotion">
                                    <swe:Quantity>
                                        <swe:uom xlink:href="urn:ogc:def:unit:revPerDay"/>
                                        <swe:value>14.20027191</swe:value>
                                    </swe:Quantity>
                                </swe:field>
                            </swe:DataRecord>
                        </sml:parameter>
                    </sml:ParameterList>
                </sml:parameters>
                <!-- METHOD DEFINITION -->
                <sml:method xlink:href="urn:ogc:def:process:SGP4:1.0"/>
            </sml:ProcessModel>
        </sml:member>
    </sml:SensorML>
```

Bibliography

[Botts, Robin, and Berthiau, 2007] SensorML Encodings of Community Sensor Models (January 08, 2008 - Version 0.1) - DRAFT

[CSM FSM07] Frame Sensor Model Metadata Profile Supporting Precise Geopositioning (June 13, 2007 - Coordination Draft, Version 1.0); Prepared for the Community Sensor Model Working Group (CSMWG)

[OGC 05-047r3] OGC 05-047r3, *OpenGIS GML in JPEG 2000 for Geographic Imagery Encoding Specification*

[OGC 06-010r6] OGC 06-010r6, *Transducer Markup Language (TML) Implementation Specification*, Version 1.0.0.

[OGC 06-009r5] OGC 06-009r5, *Sensor Observation Service*

[OGC 06-083r8] OGC 06-083r8, *OpenGIS Web Coverage Service (WCS) Implementation Specification,* Version 1.1.0

[OGC 05-103] OGC 05-103, *The OpenGIS Abstract Specification, Topic 2: Spatial Referencing by Coordinates*

[OGC 07-000] OGC 07-000, *OpenGIS Sensor Model Language (SensorML),* Version 1.0.0

[OGC 07-002r3] OGC 07-002r3, *Observation and Measurements – Part 2 – Sampling Features*

[OGC 07-022r1] OGC 07-022r1, *Observation and Measurements – Part 1 – Observation Schema*

[OGC 07-030r1] OGC 07-030r1, *OpenGIS Image Geopositioning Service (IGS)*

[OGC 07-031r1] OGC 07-031r1, *OpenGIS Image Geopositioning Metadata GML 3.2 application schema*

[OGC 07-055] OGC 07-055, *Web Coordinate Transformation Service (WCTS) draft Implementation Specification*

[OGC 07-067r2] OGC 07-067r2, *OpenGIS Web Coverage Service (WCS) Implementation Specification Corrigendum 1 (1.1.1 c1)*