

Earth remote sensing use cases for SciDB

James Frew

*Donald Bren School of Environmental Science and Management
University of California, Santa Barbara*

22 November 2008

Background

As used in this document, *Earth remote sensing* means observing the Earth with satellite-borne multispectral optical imaging sensors. *Optical* sensors measure the sunlight reflected from the Earth-atmosphere system (as opposed to, say, thermal energy emitted by the Earth-atmosphere system as a function of its temperature.) *Multispectral* sensors acquire simultaneous measurements of the reflected energy in multiple (up to hundreds of) separate spectral channels. *Imaging* sensors acquire (near-)simultaneous measurements over large areas by arranging the measurements into spatial arrays.

A simple familiar case of a multispectral optical imaging sensor is a consumer-grade digital camera, which captures a 2-dimensional array of brightness values in three spectral channels corresponding to the visible colors red, green, and blue (a wavelength range of approximately 700 to 400 nm.) Earth remote sensing uses information contained in visible and other wavelengths to derive properties of the Earth's surface.

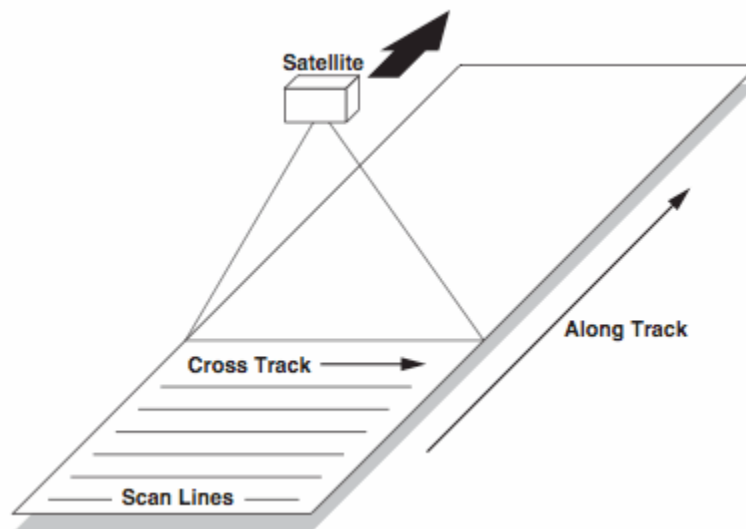
Earth remote sensing satellites are typically placed in sun-synchronous polar orbits. A *polar* orbit's ground track (path traced by the satellite's nadir point on the Earth's surface) is nearly perpendicular to the Equator; when combined with the rotation of the Earth, this allows the satellite to potentially view almost the entire Earth surface. A *sun-synchronous* orbit further constrains the satellite to pass over the Earth surface location at the same local solar time, for any location. (This is usually expressed as the time of the Equator crossing.) A constant local solar minimizes the effect of daily (but not seasonal) changes in sun (illumination) angles on the appearance of the surface.

Swaths

Because of polar orbits, most Earth remote sensors are line scanners. Rather than acquiring an entire image instantaneously, *line scanners* acquire a single line (or small group of lines) of

imagery in the cross-track direction (i.e., orthogonal to the satellite's direction of travel.) The number of lines acquired and speed of acquisition are synchronized with the satellite's orbital velocity so that the sensor has moved along-track to the location of the next line (group) when the next scan begins.

The combination of cross-track scanning and along-track motion yields an image *swath*: a continuous strip of imagery whose x-dimension is fixed by the sensor field-of-view, and whose y-dimension is potentially unbounded (but in fact is interrupted by the sensor turning off and on, the satellite passing out of range of a receiving station, etc.)



(from [HDF-EOS](#))

Note that since the Earth is rotating beneath the satellite, the swath's intrinsic x-y coordinate system does not bear a simple relationship to any Earth coordinate system. Determining the Earth coordinates of any specific swath pixel is a complex calculation involving the altitude, position, and attitude (roll, pitch, yaw) of the satellite, the optical properties of the sensor system, and the shape of the Earth. While the calculation cannot be reduced to a simple function, it is reasonable to assume that the remote sensing system's standard processing can provide the geographic coordinates (latitude and longitude) of each swath pixel.

Earth remote sensing imagery is typically not distributed to scientists as swaths since the per-pixel bookkeeping involved in utilizing swath data is prohibitive. Instead, data centers capable of swath processing generate "standard products" that memorialize certain assumptions about data usability (e.g., maximum granule size, preferred map projection, etc.) Support for swath data in SciDB would increase the usefulness of Earth remote sensing data by eliminating these intermediate products, whose assumptions often have to be "backed out" in order to support applications not considered when the products were developed.

Swath schema example

(NB: in all these examples I'll try to use the syntax of [ArrayDB](#), insofar as I understand it. Otherwise, I'll just make stuff up...)

```
CREATE swath (
  pixel BITS(nBits)[nBands],
  time DATETIME,
  location SPH_COORD_DEG (DOUBLE)
) [
  crossTrack[swathWidth],
  alongTrack[*]
];
```

Explanation:

- Raw pixel values are typically *nBits*-bit unsigned integers, where *nBits* is sensor-specific (occasionally band-specific, but that's not captured here.) I'm assuming SciDB will support this concept as a fundamental data type.
- Each pixel has an associated time-of-acquisition.
 - This doesn't necessarily have to be stored for each pixel; e.g., it may have a deterministic relationship to the start time of the scan line. I'm assuming the most general case here.
 - "Time" on satellites is often calculated with respect to a satellite-specific epoch, usually time of launch. SciDB doesn't have to store time this way, but will need to provide functions that convert between satellite and absolute time.
- Each pixel has an associated geographic coordinate (latitude and longitude in some well-known geographic coordinate system; e.g. WGS84.) I'm assuming `SPH_COORD_DEG (DOUBLE)` can handle this.

Swath query examples

Select all *pixels* that whose locations lie within the geographic footprint of *object* (array, geometric feature, etc.) The result set is a 1D array. This form is useful for calculating aggregates.

```
SELECT *
FROM swath
WHERE WITHIN(location, object);
```

Select all *array rows* containing one or more pixels whose locations lie within the geographic footprint of *object*. The result set is a 2D array whose first dimension is *swathWidth* and whose second dimension is determined by the `WITHIN` condition. This form is useful for

extracting swath subsets for subsequent processing.

```
SELECT crosstrack[*]
FROM swath
WHERE WITHIN(location, object);
```

Projections

While swaths are useful for accommodating raw remote sensing data, they are not the preferred data structure for analysis (although that may change if SciDB makes swath support ubiquitous.) Instead, Earth science takes place in projected coordinate systems, both to accommodate historic data, and to simplify analyses (i.e. by assuming a unit stride in the array corresponds to a constant distance on the Earth's surface.) SciDB will need to transparently support projected arrays.

Projection schema example

I'll assume that SciDB keeps track of projection information in some common subschema, analogous to [\[OGC\]](#). A *spatial reference system ID* can thus server as shorthand for particular parameterization (e.g., UTM Zone 11) of a particular projection (e.g., UTM) of a particular geographic coordinate system (e.g., NAD 1983.) Thus:

```
CREATE image (
  pixel FLOAT[nbands]
) [
  easting[PROJECTED_DIM(srsId, "E", eMin, eDelta, eSize)],
  northing[PROJECTED_DIM(srsId, "N", nMin, nDelta, nSize)]
];
```

creates a 2D array of multivariate pixels whose dimension *easting* (*northing*) is defined as the *E* (*N*) dimension of coordinate system *srsId*, with index values beginning at *eMin* (*nMin*) and incrementing by *eDelta* (*nDelta*) for *eSize* (*nSize*) rows (columns.)

Projection query examples

Select a rectangular subset of an image, specified in projected coordinates.

```
SELECT *
FROM image[350000..360000][1200000..1300000];
```

Combine projected images with commensurate dimensions (commensurate here being defined as: same spatial reference system, same delta values, and minimum coordinates that differ only by integral delta values.) The resulting array may be either ragged, or filled out with `NULL`

values to the bounding box of the non-NULL values.

```
SELECT *  
FROM image1 {AND,OR,XOR} image2;
```

References

ArrayDB

A Sketch of ArrayDB (2008) (where does this live?)

HDF-EOS

HDF-EOS Interface Based on HDF-5, Volume 1: Overview and Examples (2008)

<http://edhs1.gsfc.nasa.gov/waisdata/rel7/html/175emd001r5.html>

OGC

OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option (2006) <http://www.opengeospatial.org/standards/sfs>