DESIGN AND IMPLEMENTATION OF AN OBJECT-ORIENTED PHOTOGRAMMETRIC TOOLKIT

Chris McGlone, Ph.D.
Loral Fairchild Systems, Inc.
300 Robbins Lane
Syosset, NY, USA 11791
ISPRS Commission II

ABSTRACT

This paper describes an object-oriented softcopy photogrammetric toolkit, designed to allow the use of different types of images, such as frame, panoramic, strip, and satellite, without modifying applications code. Addition or modification of new sensor models requires no modification of existing applications software. The orientation module can simultaneously perform a least squares orientation and point coordinate determination on images of different geometries. Written in C under the UNIX operating system, the toolkit is portable across platforms and has been integrated into softcopy image exploitation systems.

Key Words: Digital systems, Photogrammetry, System design, Workstation

INTRODUCTION

An ideal softcopy image exploitation system would be able to exploit images with greatly differing geometries—line scanners, strip (pushbroom), panoramic, and satellites, as well as digitized film—singly or in mixed combinations, with rigorous results. This goal has yet to be completely attained for systems currently in use, due to conventional software architectures which typically require modifications to the entire body of code each time a new sensor model is added and which require every application to have explicit knowledge of all types of images. As a consequence, software production and maintenance costs are high.

The photogrammetric toolkit has been designed as an answer to these problems. It consists of a set of sensor models, arranged in a class hierarchy by imaging geometry, and a set of application routines designed to work with generic sensor model objects without knowledge of the internal workings of the model or geometry of the sensor. The addition or modification of sensor models has no effect on other sensor models or on the applications routines.

To ease integration of the toolkit into various systems, the sensor models and applications do not directly communicate with the user interface; instead, they rely on the system in which the toolkit is embedded to pass commands and data and to return results to the operator. Since the toolkit is not rigidly tied to a particular user interface, development and maintenance costs can be amortized across many projects.

*Currently at Digital Mapping Laboratory, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 15213.

DESIGN GOALS

Requirements were analyzed in terms of necessary photogrammetric applications, accuracy specifications, and software engineering considerations. The final requirements were:

- The system must handle different types of sensor geometries, including frame, panoramic, strip, line scan, satellite, and radar.
- The system must be able to use, input, and output results in standard coordinate systems—geodetic (latitude, longitude), geocentric, UTM, and local vertical.
- The system must work with a flat, spherical, or elliptical description of the earth, with specifiable ellipsoid parameters.
- The system must not contain hidden approximations which degrade the accuracy of the solution or which are specific to a certain sensor or imaging scenario.
- The system must support standard photogrammetric operations such as positioning, 2D and 3D measurement and multiple-image resection and intersection solutions. The addition of other applications must be done with a minimum of effort.
- It must be usable as a component in applications such as orthophoto generation or perspective transformations and in research applications.
- The system must be easily transportable between systems. Instead of machine-specific and interface-specific routines, widely-accepted software standards such as the C language, the Unix operating system, and the Motif user interface must be used.

Design of the toolkit was influenced a great deal by work in object-oriented graphics programs, especially the M-BLORB modeling package implemented at the Schlumberger Palo Alto Research Center by Kurt Fleischer [Fleischer, 1987].

One of the few published attempts to deal with different image geometries in photogrammetric systems was described in [Molander, *et. al.*, 1987]. Their approach applied the same mathematical model to frame, panoramic, and strip imagery, treating all images as composed of framelets taken at discrete time intervals. The orientation parameters for each framelet were determined from time-indexed lookup tables. While this approach followed some of the philosophy of object-oriented design, it did not allow for the use of completely different orientation methods for sensors such as radar which might require them.

TOOLKIT STRUCTURE

The toolkit is divided into modules, each of which is concerned with an object or a related set of functions (Figure 1). The fundamental objects of the toolkit are worlds, points, cameras, and images. Each sensor model consists of a camera and an image object, related to its place in the hierarchy of sensors, along with functions which perform the generic operations used by the applications. The applications are written solely in terms of these generic operations. A package of utilities for operations such as coordinate transformations and units conversion is used in writing the sensor models and applications and by the external system.

The world object contains the world parameters for use in computations, such as ellipsoid parameters, and the choices for input and output coordinate systems, input and output units, tick mark spacings, etc.

A point object represents a single point in the world. World coordinates in the various coordinate systems are stored as calculated or input, with coordinate conversions performed as needed. Image coordinates on each image the point appears on are also stored, along with stochastic information.

Cameras and images are separate objects, although they are treated together. The camera object contains information about the sensor, while the image object refers to the specific event when an image was made and contains position and orientation information. A sensor model is defined by a camera object and an image object, along with the relevant functions. These functions can be inherited from other sensors or specially written.

Sensors are arranged in a shallow hierarchy, (Figure 2) with the root of the tree being a proto-image, the next level being sensors with generic geometries (frame, panoramic), then the next being specific sensor types. Generic functions can be inherited down the tree or specified for each sensor.

Generic functions (or methods) for sensor models include:

- `set_values`. Sets the orientation and position values for the image.
- `show_values`. Displays the current image values.
- `image_to_world`. Transforms a set of image coordinates into a set of world coordinates in the specified coordinate system.
- `world_to_image`. Transforms a set of world coordinates into the corresponding image coordinates.
- `read_image_from_file`, `write_image_to_file`. These functions handle the input and output of image information.
- `image_to_subimage` and `subimage_to_image`. Perform the transformation between displayed subimages or reduced-resolution images and the original image.
- `calc_partials`, `update_parameters`. Used in the resection calculation to calculate partial derivatives and to update the parameter approximations.

Lens distortion and atmospheric refraction correction functions are provided as utility routines, with each sensor model using the correct form with its own parameters.

Application routines call the generic functions without knowing the inner workings of the sensor models. While this sometimes requires writing the application in a way that may be less intuitive, once done it does not have to be modified for new sensor types. For example, for most sensors, calculating the size of a pixel on the ground can be done simply using the focal length, the detector size on the imaging array, and the distance to the ground. However, some electro-optical sensors oversample or undersample the scanned output so that a pixel does not correspond precisely to the detector size. This application was written using the `image-to-world` function called for adjacent pixels, then calculating the distance between their projections in ground coordinates.

IMPLEMENTATION

The toolkit is currently implemented in the C programming language, with the basic objects represented by structures. The generic functions such as `image_to_world` and `world_to_image` are attached to slots in the structures. Pointers to image objects are passed as pointers to void; structure slots are addressed by casting the pointer to `proto_image` or to the specific type of image in functions internal to the sensor model.

The sensor models and application routines do not assume any user interface or image display hardware or software; all user communication and image handling is the responsibility of the system in which the toolkit is embedded. Input parameters and image coordinates selected by the operator are placed in the appropriate structures and passed to the applications functions. Calculated results such as coordinates are returned in the objects passed to the routines, and are also formatted as strings returned from the applications. By separating the interface issues in this way, the system can be ported between systems without modification of the code.

The toolkit is currently running on Sun workstations under Sunview, and on the EO-LOROPS Ground Exploitation Station (VAXstation based) with the interface written in DECWindows. The next target platform is the Silicon Graphics with the interface to be written in Motif, an X-Windows based user interface.

Sensor models

Models describing frame mapping cameras and the GD HIAC frame camera have been implemented, along with a model for the KA-93 film panoramic camera and the EO-LOROPS (Electro-Optical—Long Range Oblique Panoramic) electro-optical sector-scan panoramic camera. Models to be added include the ATARS (Advanced Tactical Airborne Reconnaissance System) F979L and F979M strip sensors built by Loral Fairchild, SPOT and LANDSAT satellite sensors, IR line scanners, and synthetic aperture radar.

Applications

Current and future applications routines include:

- *Calculation of ground and image coordinates.* The system calculates ground coordinates from given image coordinates or calculates image coordinates from given ground coordinates. Ground coordinates can be input or output in geodetic (latitude-longitude), UTM, geocentric, or local vertical coordinate systems.
- *Monoscopic height calculation.* If the top and bottom of a vertical object are visible, an approximate height can be calculated and output in the specified linear units.

- *Horizontal distance calculation.* Horizontal distances are output in user-specified linear units.
- *Area calculation.* The area within an arbitrary polygonal region is calculated and output in user-specified area units.
- *Coverage calculation.* The coordinates of the corners and center of the image can be calculated and displayed in any coordinate system.
- *Overlay coordinate tick marks.* Tick marks in lat-long or UTM coordinates, with user-specified spacings, can be overlaid on the image.
- *Pixel information.* The system will calculate the ground sampled distance in x and y, the slant and ground ranges to the point, and the depression angle at a given point in the image.
- *Coordinate and units conversion.* The system can work with geodetic (latitude-longitude), geocentric (earth-centered cartesian), Universal Transverse Mercator (UTM), and local vertical coordinate systems. All internal calculations are done in geocentric or local vertical coordinate systems, with coordinate conversions done automatically as required. The operator can input coordinates or obtain output results in whichever coordinate system he desires. A separate function allows the operator to input coordinates and convert them to any other system.

  Input and output units are also user-specifiable. Angles may be given in radians, decimal degrees, degrees-minutes-seconds, or grads. Linear units are meters, feet, kilometers, miles, and nautical miles, while area units are square meters, square feet, acres, and hectares.
- *Simultaneous orientation determination.* In many cases the position and orientation data for an image is unknown or not known accurately enough. A resection solution must then be run using ground control points with known coordinates to obtain better values for the orientation parameters. In the converse problem, intersection, the coordinates of ground points must be determined from two or more known images. A third problem is performing relative orientation between two or more images, without any known ground coordinates. In the most general case, a mixture of known, unknown, and partially known points and images may be adjusted.

  A general simultaneous orientation solution has been implemented in the toolkit which addresses these separate problems using the same program. The type of solution is implicitly selected by the choice of weights for the image coordinates, ground coordinates, and image orientation parameters. The orientation solution can use multiple images from any modeled sensor or combination of sensors. For instance, if overlapping panoramic, frame, and satellite imagery of a given area exist they can be combined in the same solution, thereby obtaining better results than if each had been resected separately and requiring fewer total control points. Tie points, common points without known coordinates identifiable on more than one image, can also be included to strengthen the adjustment.
- *Positioning precision estimation.* The precision (standard deviation) of ground positions or measurements is often of interest. Error propagation results are available as an after-product of the orientation solution.
- *Image point geometry and precision simulation.* It may be desirable to estimate the location of ground points in an image or the precision of ground point positioning before an actual mission is flown. The image point simulator takes point world coordinates and generates sets of image coordinates which can be used in the resection procedure to estimate the precision of the image and point location for that particular geometric configuration.

Interactive multiple image applications for the toolkit, such as stereo display and exploitation or coordinated viewing of non-stereo imagery, depend upon the user interface and the image display system. The interface driver is responsible for accepting operator input in mono or stereo, calling the proper applications to calculate image or world coordinates, and displaying the results.

Outline of a typical application

A typical application begins by creating the image object or reading it from a file. The routine which creates the object is the one routine in the system which must be modified when a new sensor model is added. A "create-yourself" method will not work, since the object does not exist until it's created but can't create itself until it exists. To get around this, the creator/reader can either create an object of the specified type or reads the type as the first field of the file containing the sensor's parameters. Once the object is created, the `set_values` and `read_image_from_file` methods are available.

An interesting application is the resection module, which works with multiple images of mixed types. Resection begins by allocating the solution matrices, which must be dimensioned according to the total number of unknown orientation parameters. This total is obtained by adding the `nbr_parameters` fields of each object. If approximations for point coordinates are needed, they are generated using the `image_to_world` function for that sensor. The least squares solution requires the generation of partial derivative matrices with respect to the orientation parameters for each image. The `calc_partials` method returns a matrix of dimension 2 by `nbr_parameters`, which is used to form the portion of the normal equations corresponding to that image.

Once each image has added its contributions to the normal equations, the equations are solved and parameter corrections are obtained. Each image's parameters are updated by calling its `update_parameters` method, which adds the deltas, recalculates the orientation matrices, and performs any other required updates for that sensor. The resection solution iterates until the solution converges, when error propagation can be calculated if desired.

FUTURE DIRECTIONS

Implementation, integration, and testing has confirmed the basic approach, revealed some problems, and shown further extensions and applications of the concept. Some of these are discussed below.

Issues revealed

The toolkit has thus far fulfilled its design goals. The first integration into a deliverable image exploitation system went well. Two sensor models used in generating demonstration results for a proposal were quickly written, tested, and used.
An unfortunate limitation on the toolkit was the requirement that it be written in C, necessitating the simulation of object-oriented facilities that other languages such as C++ already have. Languages such as C++ or Lisp with CLOS would have provided a much more powerful and efficient development environment.
Execution efficiency has not been a problem, given that the toolkit has needed to work only at interactive rates in response to operator input. If it were used as part of an automated system such as an orthophoto generator some optimization would be desirable. One of the main efficiency issues is the "granularity" of the methods—allowing external callers to use only top-level functions, such as `image_to_world`, makes the interface clean but may require unnecessary re-computation of intermediate results when a large number of points from the same image are calculated. To optimize execution time, some methods may have to be split into sub-methods.
As noted before, the inheritance tree for sensor classes is relatively shallow (Figure 2). Multiple inheritance would have been useful, but was not implemented.
There is no reason that single-image applications should necessarily be written independent of the sensors. These could better be written as methods for each particular sensor. Only multiple-image applications need to be blind to the internal workings of the sensor models, since they may have to deal with two different types of sensors at the same time.

Potential enhancements

An interesting possibility is the inclusion and utilization of "sensor models" of things that are not normally considered sensors. For instance, a raster image of map can be considered an image with the projection functions defined by an orthographic projection. Orthoimages can be treated the same way.
With the addition of functions to get and set the image value (intensity) at a given location, the toolkit could be used as the basis for a image geometry transform system. Given an input image and an output image of any specified geometry, the program would consist of a simple loop over the coverage of the output image:

1. For each pixel in the output image, use `image_to_world` to calculate its world coordinates.
2. For each set of world coordinates calculated above, use `world_to_image` on the input image to get the input image coordinates corresponding to it.
3. `Get_image_intensity` on the input image at the calculated image coordinates.
4. `Set_image_intensity` on the output image to the intensity of the input image.

Boundary conditions and sampling considerations would have to be taken care of, but otherwise the implementation is trivial.
An "image" consisting of a computer-graphics representation of a scene could also be used, either a previously rendered scene or it could be rendered as required. The `image_to_world` function would be defined as intersecting the ray defined by the specified pixel with the models or background. The `get_image_intensity` function would be defined as either getting the rendered intensity values or else performing the rendering using a light source specified by the world object. This type of "image" would be ideal for perspective transform operations.
Applications in non-photogrametric systems have also been considered. For instance, sensor performance prediction models often require geometric information about the sensor. Use of the toolkit would allow the performance model to be independent of knowledge of the particular sensor.

SUMMARY

The photogrammetric toolkit, built using object-oriented programming techniques and industry standard software packages, can be easily integrated into image exploitation workstations and provides rigorous modeling of a wide variety of sensors. New sensor types can be added with a minimum of effort and with no modification of existing applications code, contributing to low software maintenance costs and to system flexibility.

References

[Fleischer, 1987] Fleischer, K., "Implementation of a Modeling Testbed," *Association for Computing Machinery SIGGRAPH 1987 Course Notes*, Volume 14.

[Molander, *et. al.*, 1987] Molander, C., Moraco, A., and Houck, D., "A Numerical Photogrammetric Model for Software Applications", *Proceedings of the American Society for Photogrammetry and Remote Sensing Annual Convention*, Vol. 2, 1987, pp. 284-292.

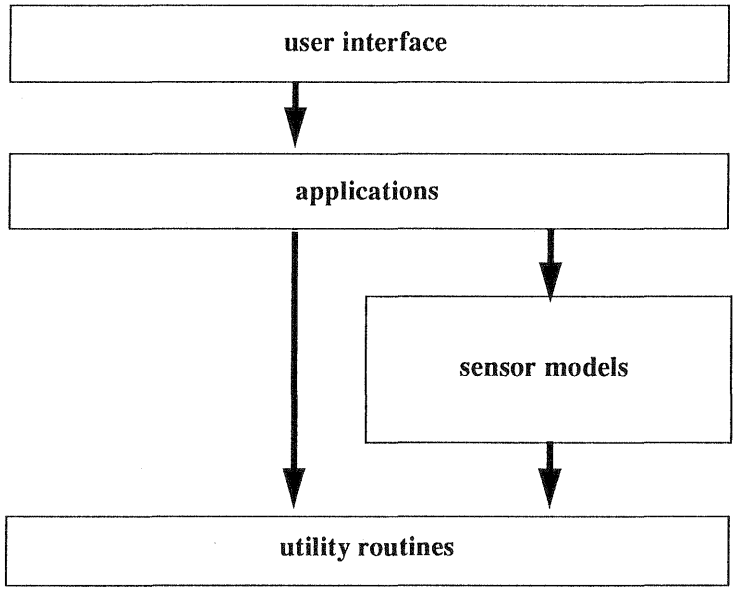

Figure 1: Toolkit module structure.

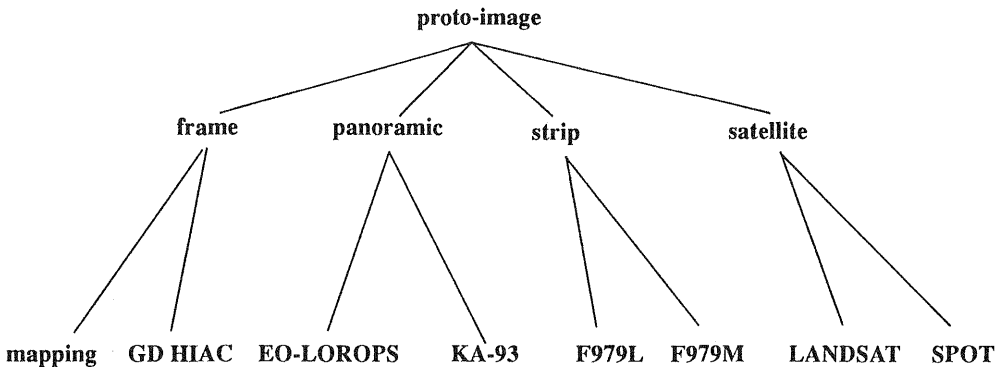

Figure 2: Sensor model hierarchy.