TEACHING PHOTOGRAMMETRY ALGORITHMS BY DEDICATED SOFTWARE

Vittorio Casella

Università di Pavia
Dipartimento di Ingegneria Edile e del Territorio
Via Ferrata, 1 - 27100 Pavia - Italia
email: casella@ipv36.unipv.it

Commission VI, Working group 3

KEYWORDS: Photogrammetry, Teaching, Algorithm, Least squares

ABSTRACT

A description is given of a set of programs conceived and written to teach analytical photogrammetry algorithms. Programs'aim is to make the students understand how these algorithms work and which role the various parameters play. Programs will be freely distributed.

1 INTRODUCTION

Information technology can greatly enhance teaching techniques. In photogrammetry teaching, in particular, it permits every student to elaborate his own data (i.e. measured by himself) directly.

The paper describes a set of programs written in the Matlab environment, which are able to solve the basic analytical photogrammetry problems and which have been conceived and implemented to improve photogrammetry teaching at the University of Pavia. The paper also describes the way they are used.

2 THE *RAISON D'ETRE* OF THE PROJECT

It is well known that it is difficult to make students fully understand some specific details of photogrammetry theory and algorithms only by means of the traditional, ex-cathedra, lessons. The Photogrammetry teaching staff at the University of Pavia has become aware, over the last two years, of the necessity of having specific software able to calculate the photogrammetric orientations and able to successively plot unknown points. All this, with the aim of allowing the students to elaborate data measured by themselves in an *autonomous* and *individual* way. The programs described are the first answer to these expectations: indeed the work is ongoing and, at the end of the paper, new features will be described that it would be useful to add in the near future.

3 MAIN FEATURES OF THE PROGRAMS

At the moment programs are able to solve interior, relative and absolute orientations and are also able to plot, after the orientation procedures have been finished. All the routines have the same philosophy: the user somehow measures on images (photographs or digital files) points instrumental coordinates (with an analytical plotter, a digital plotter, a simple image processing software or a digitizing tablet), and writes them in a textual file; then the suitable software procedure is called: it reads data from the files, works them out and displays results on the screen, together with accurate statistical analysis and graphical representation; to finish, it writes results on the disk so that they are suitable to the following orientation step.

Let's briefly consider an example. To complete the interior orientation of one photograph, it is necessary that the user measures and records on a file the actual fiducial marks coordinates and writes to another disk file theoretical fiducial marks positions, according to the camera calibration certificate. Then the user has to call the *interior orientation calculation program*, specifying data files and which kind of coordinate transformation (there are few different kinds, which will be described later in the paper) has to be applied. The procedure evaluates the transformation parameters, records and shows them, together with quality parameters and residuals. There is also an *interior orientation application program* that, given the instrumental coordinates of a set of points, provided the interior orientation parameters, transforms instrumental coordinates into image coordinates.

The programs are highly *modular*, so they allow the user to focus exactly the role of the various steps of the pho-

togrammetric procedure and the role of the different parameters involved; moreover they are *open*: it's easy to read, modify or improve them. This is not always possible with commercial software, which is, by its own nature, closed and consequently hides many details of the internal algorithms from the user .

4 HOW PROGRAMS ARE USED

The students are required, first of all, to make measurements at our Geomatica Laboratory; they are subdivided into groups of five or six and they have to take measurements of a certain number of marked points (fiducial marks, tie points, ground control points, points to be measured) with three different instruments solely used as collimators: an analytical stereoplotter, a digital low-cost stereoplotter and a digitizing table; the instrumental coordinates are written in files that have to be processed later.

Completely different instruments are used to make measurements, with completely different working principles and, of course, with very different quality levels. All this helps us to underline that analytical photogrammetry algorithms are completely separate from analytical stereoplotters (this is obvious, of course, but students often get confused, especially if lessons and practices *de facto* equate the two things). This allows also to show to the students that photogrammetry procedures have a statistical nature, so that if good data are put in, they give back good results, and viceversa.

For the subsequent processing, students and staff have access to specialised laboratories, each with twenty computers, where the students, subdivided into groups of not more than two or three, are guided along the data processing and into the critical examination of the results. In this way, we offer the students the possibility of seeing results as they arise, step by step through the whole procedure, together with the possibility of understanding, as a consequence, the role and the effects of each different phase and of each parameter involved. Moreover, using data coming from different measurement instruments, and characterised, of course, by different accuracy levels, allows the main orientations quality indicators to be recalled.

It is also very important and useful that the students have access to the computer laboratories also individually, out of the teaching time. In this way they are allowed to freely use the already known procedures and to better understand their functioning. It is to be underlined that the students can access directly the program listings, so they can analyse the algorithms details or even modify them, if this helps them to understand.

5 DESCRIPTION OF THE SOFTWARE

Rather than a detailed description of the programs, the following section will focus on their global architecture and will outline main procedures classes, main function prototypes and the environment in which programs are written, Matlab. A detailed handbook of the software tools is currently carried out and will be distributed with the software, as soon as possible.

Programs will be outlined grouped by categories: there are two functions' groups that are needed to work out analytical photogrammetry problems, but that have wider interest and usefulness: they are the least squares function group and the coordinate transformation function group. For these two groups the description will be a bit more detailed, so as to show their capacities. For other programs, strictly focused on traditional, well known, analytical photogrammetry algorithms, the description will be synthetic.

5.1 Least squares functions

It has been written a universal least squares solver, able to solve any problem, provided that the user builds a dedicated function that calculates the deterministic model.

5.1.1 Theoretical background. The programs that are currently available manage only the parametric deterministic model

$$\mathbf{Y} = \mathbf{AX} + \mathbf{a} \tag{1}$$

where $\mathbf{Y}$ is an m random variable that we are able to measure, and $\mathbf{X}$ is an n random variable for which we want to give an estimation of the mean value, based on an extraction (a measurement) of $\mathbf{Y}$, usually indicated by $\mathbf{Y}_0$. This is not a big limitation because it is known that also the most general linear deterministic model

$$\mathbf{DY} = \mathbf{AX} + \mathbf{d} \tag{2}$$

can be reformulated in terms of a parametric model problem. The user has also to supply the stochastic model, that is the structure of the variance-covariance matrix of the $\mathbf{Y}$ random variable. The word *structure* recalls that, writing the variance-covariance matrix in the form $\mathbf{C}_{YY} = \sigma_0^2\mathbf{Q}$, the matrix $\mathbf{Q}$ has to be given by the user, while the coefficient σ_0 will be estimated by the least squares solution.

The routines calculate the well known solutions for the estimated parameters

$$\mathbf{X} = \left(\mathbf{A}'\mathbf{Q}\mathbf{A}\right)^{-1}\mathbf{A}'\mathbf{Q}^{-1}\left(\mathbf{Y}_0 - \mathbf{a}\right) \tag{3}$$

and for the estimated observations

$$\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{a} \tag{4}$$

and for the overall quality parameter

$$\sigma_0 = \sqrt{\frac{(\mathbf{Y} - \mathbf{Y}_0)'\mathbf{Q}^{-1}(\mathbf{Y} - \mathbf{Y}_0)}{m - n}} \tag{5}$$

The routine also estimates the variance-covariance matrixes for both parameters and observations, respectively

$$\mathbf{C}_{XX} = \sigma_0^2\left(\mathbf{A}'\mathbf{Q}\mathbf{A}\right)^{-1} \tag{6}$$

$$\mathbf{C}_{YY} = \mathbf{A}\mathbf{C}_{XX}\mathbf{A}' \tag{7}$$

In case of non linear problems

$$\mathbf{Y} = f(\mathbf{X}) \tag{8}$$

solution is reached iteratively by a sequence of linear approximated problems: the user has to supply an approximated solution $\mathbf{X}_0$. Iterations are stopped when they give no more significant enhancement to the solution.

5.1.2 The least squares solver engine. The engine routine is `ls_calc`, and has the following prototype:

```
[EstPar, EstObs, Quality] =
ls_calc(DetMod, Obs, Opt, OptData)
```

Let's examine briefly input arguments:

- `ModDet` is a structure (sources have already been updated to the fifth release of Matlab) containing two fields: `Name`, the name of the deterministic model function, and `Xa`, the aproximated parameters vector, if any;
- `Obs` is a structure containing the fields `Y0`, the observations vector, and the matrix of observations variance-covariance, `Q`;
- `Opt` is a structure containing some parameters that the user can fix so as to tune program behaviour. It contains the following fields: `ls_Eps`, related to iterations stop condition; `ls_IterMin`, which fixes the minimum iterations number (the user could choose a value greater than one, for testing reasons); `ls_IterMax`, the maximum number of iterations, to prevent infinite loops; `ls_Verb`, which controls the verbosity of the screen and file echo: it is possible to choose to have (i) no echo, (ii) a screen echo of the essential parameters of the running process, (iii) a file dump of the process, (iiii) a complete and long file dump, including the condition number of the deterministic model matrix A.
- `OptData` is a structure which isn't used directly by the engine, but it is passed to the deterministic model function instead. It allows to pass to the deterministic model function data which are neither in the observations vector $\mathbf{Y}_0$, nor in the approximated values vector $\mathbf{X}_0$, but that are, nevertheless, necessary to calculate the matrixes $\mathbf{A}$ and $\mathbf{a}$ of the deterministic model.

Let's examine output arguments now:

- `EstPar` is a structure containing two fields: `Xs`, the estimated parameters vector and its variance-covariance matrix, `Cxx`;
- `EstObs` is a structure containing the fields `Ys` of the estimated observations, the matrix `Cyy` of its variance-covariance matrix, and the `Nu` vector of the residuals $\mathbf{Y} - \mathbf{Y}_0$;
- `Quality` is a miscellaneous structure containing, among the others: `Sigma`, that is the value of the quantity indicated by the eq. (5) and `IterNum` the number of the iterations executed.

5.1.3 Final remarks on the least squares engine. As already underlined, the engine is able to solve any problem, provided that the user builds the deterministic model function. This function must have a name beginning with the prefix '`dm_`' and has to give back to the calling function the deterministic model matrixes $\mathbf{A}$ and $\mathbf{a}$.

It is noticeable that, thanks to the structures data types, that have been introduced in Matlab only recently, it will be possible in the future to modify functions' behaviour without changing the prototypes. This will allow to change the software without being necessarily obliged to change the calls to the engine, spread into many different programs.

5.2 Point class

The current version of the package takes full advantage of the object oriented features of Matlab's language. There is a class called `Point3D`, which has the role of recording the three dimensional coordinates of a set of points: for each point it is possible to store a name, the three coordinates and, if any, the variance-covariance matrix.

5.3 Coordinate transformation functions

Coordinate transformation functions constitute another self-consistent set of functionalities. They have been written taking advantage of the object oriented features of Matlab's language. Each coordinate transformation is a class, with many overloadable methods.

Each of these classes contains, among the others, the parameters values, their variance-covariance matrix, if any, and the `Versus` flag, assessing if the transformation has to by applied directly or backward.

5.3.1 Direct application of coordinates transformations. One of the methods of coordinates transformation classes (let `CT` indicate any of them) is `apply`, which transforms a `Point3D` class instance according to the rules specified by `CT`. If variance-covariance matrixes of the to-be-transformed points are known, the method estimates also the stochastic behaviour of the transformed coordinates.

If `CT` is an instance of a certain coordinate transformation class and if `Pt` is an instance of the `Point3D` class, the following instruction

```
NewPt=apply(CT, Pt)
```

transforms the coordinates contained in `Pt` according to `CT` and stores the results in `NewPt`.

The package offers three different Planar Coordinate Transformations, `PCT` starting from now, with the functional form

$$\mathbf{P}_u = \mathbf{D}\mathbf{R}\mathbf{P}_x + \mathbf{T} \tag{9}$$

where **R** is the rotation matrix

$$\mathbf{R} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{10}$$

while **T** is the translation vector, with the third component equal to zero, due to the planarity of the transformation. **D** is the 'deformation matrix' which changes depending on the transformation type; **D** has, in the case of `PCT3`, the following shape

$$\mathbf{D} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{11}$$

while, in the case of `PCT4`, the matrix **D** has the form

$$\mathbf{D} = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{12}$$

and in the case of `PCT5`, **D** is

$$\mathbf{D} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{13}$$

The package currently offers only one three dimensional coordinate transformation, the well known seven parameters transformation (called `SCT7`), having the form

$$\mathbf{P}_u = \lambda\mathbf{R}\mathbf{P}_x + \mathbf{T} \tag{14}$$

where **R** is the usual three dimensional rotation matrix

$$\mathbf{R} = \mathbf{R}_\omega\mathbf{R}_\varphi\mathbf{R}_\kappa$$

$$\mathbf{R}_\omega = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{pmatrix}$$

$$\mathbf{R}_\varphi = \begin{pmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{pmatrix} \tag{15}$$

$$\mathbf{R}_k = \begin{pmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

5.3.2 Parameters estimation of coordinate transformations. The method `calc` makes a least squares estimation of the parameters of a certain transformation, provided the coordinates of a sufficient number of points respect to two different reference systems.

The `PCT4` case is solved exactly, thanks to the well known linearizing variable transformation. `PCT3`, `PCT5` and `SCT7` are solved iteratively, starting from approximated values. Such values must be passed to the function by the user or, in the case of `PCT3` and `PCT5`, can be estimated by the method itself by a preliminary `PCT4` estimation.

If `PointUV` and `PointXY` are two instances of the class `Point3D` containing the coordinates of some points referred to two different reference systems, the instruction

```
ECT=calc(CT, PointUV, PointXY)
```

estimates the parameters of a coordinate transformation and stores them in the `ECT` object, belonging to the `CT` class. The `CT` object must be an instance of the chosen

transformation class and fixes the transformation type and the approximated solutions, if they are needed. The CT object can be void if approximated parameters are not necessary (PCT4 case) or if they are unknown: in this case the `calc` method estimates such approximated parameters in the best possible way.

5.4 Interior Orientation routines

The interior orientation programs mainly uses the functions and the classes already outlined. They estimate the parameters of a PCT which transforms the instrumental reference system into the image reference system, on the basis of nominal and measured fiducial marks coordinates. There are CTF objects, containing camera certificate information, and IOD (Interior Orientation Data) objects, able to store measured fiducial marks positions of one photograph. The principal class is IOEP (Interior Orientation Estimated Parameters) which stores the estimated parameters, together with links to the processed data object and many other data, useful for statistical quality analysis. Among IOEP class methods, there are `calc`, with the following prototype

```
IOEP=calc(IOEP, CTF, IOD)
```

which estimates the parameters and give graphical visualisation of the residuals, and `apply`, which transform instrumental coordinates into image ones.

5.5 The relative orientation program

The relative orientation program, `ro_calc` has the following prototype

```
ROEP= ro_calc(ROEP, IOEP, IOEP, ROD)
```

where the two passed IOEP class instances contain the interior orientation parameters of both involved photographs; the fourth passed argument is an instance of the ROD class (Relative Orientation Data) which contains, mainly, the instrumental coordinates of tie points on both left and right image. The returned argument, a ROEP (Relative Orientation Estimated Parameters) class instance, is somehow analogous to the IOEP class and contains the orientations values of both image, respect to the model arbitrary reference system. The `ro_calc` program uses the *coplanarity* condition to determine relative *symmetric* orientation parameters: the algorithm is able to linearize the observation conditions everywhere, although it is not able to determine the approximated solutions, which are chosen in the usual way, with all the angles null.

There is also a `plot` function, which determines the three dimensional coordinates (model coordinates or object coordinates) intersecting homologous lines; it uses a geometrical approach, there is no adjustment indeed, and only three of the four available equations are used.

5.6 The absolute orientation program

The absolute orientation program, `ao_calc`, has the following prototype:

```
AOEP= ao_calc(AOEP, ROEP, AOD, GPD)
```

where the second passed argument, a ROEP class instance, contains the relative orientation parameters; the AOD (Absolute Orientation Data) class instance contains, mainly, the instrumental coordinates of control points on both left and right image and GPD class instance records ground coordinates of control points. The program estimates the parameters of a SCT7 transformation; as in the relative orientation case, the program is able to linearize equations everywhere, but isn't able to find a good approximated solution, that must be passed with the first argument; if this one is void, the program start with the standard approximated solution, with the angular values equal to zero.

Obviously the returned AOEP (Absolute Orientation Estimated Parameters) class instance contains the exterior orientation values of the images, together with their variance-covariance matrix and other quality indicators.

5.7 The programs language

Programs are written in the Matlab language, so they need that environment to be executed, and this could be an obstacle to the diffusion of the procedures. On the other hand, Matlab allows development times much shorter than a standard language, such as C language, does. Moreover, Matlab procedures are platform independent: the same programs can run in Matlab for Windows and in Matlab for Unix.

5.8 Programs documentation and availability

As already mentioned, there isn't yet a programs handbook; however, all the functions are documented, although in italian. Programs will be freely distributed, so a package copy will be sent to whoever will ask them.

5.9 Further developments

Among all the many things that it would be fine to add to the packet, there are certainly the bundle adjustment and the digital images support. The first step in the digital field will be the construction of a digital collimator, so that the above described seminars will only need one computer for both measurements and further processing: digital photogrammetry has great potential also in the teaching field.

6 CONCLUSIONS

Information technology can greatly contribute to making teaching effective and up-to-date, on the condition that the necessary software instruments are acquired: this requires at the least a big time investment; for this reason it has been decided to freely distribute the programs. The author and the whole photogrammetry staff of the University of Pavia hope that this is the starting point of a deeper collaboration in the field of teaching problems between teachers and researchers from different Universities, based on exchanges of experiences, materials and instruments.

7 REFERENCES

Benciolini, B., 1989. Modelli di Compensazione. In: Progettazione e Ottimizzazione del Rilievo Topografico e Fotogrammetrico di Controllo, CISM, Udine, Italy.

Benciolini, B., 1990. Modelli Analitici di Base della Fotogrammetria. In: Dall'Analitico al Digitale; Nuovi Sviluppi della Fotogrammetria applicata all'Ingegneria, CISM, Udine, Italy.

Kraus, K., 1994. Fotogrammetria: Teoria e Applicazioni, Vol I, trad. Sergio Dequal, Torino. Libreria Universitaria Levrotto & Bella, Torino, Italy.

Mikhail, E.M., 1976. Observations and Least Squares. IEP, New York.

Mussio, L., 1984. Il calcolo delle Trasformazioni Piane Elementari. In: Ricerche di Geodesia Topografia e Fotogrammetria, CLUP, Milano, Italy, Vol IV.

Sansò, F., 1989. Il Trattamento Statistico dei Dati. Città Studi, Milano, Italy.