XIV Congress of the International Society for Photogrammetry
Hamburg 1980

COMMISSION III

WORKING GROUP III-4

PRESENTED PAPER

Rune Larsson

Department of Photogrammetry
Royal Institute of Technology
Stockholm, Sweden

EFFICIENT DESIGN OF A SYSTEM FOR SIMULTANEOUS ADJUSTMENT OF PHOTOGRAMMETRIC AND GEODETIC OBSERVATIONS AND ADDITIONAL PARAMETERS

ABSTRACT

The design of a general photogrammetric system to be implemented on a mini-computer is outlined. Existing systems for this task are often designed for the big computer and rely heavily on the fast multi-channel direct-access architecture of these. These programs can often, however, be implemented on smaller computers, but then the speed of execution will be limited by conflicting demands for fast direct-access secondary storage.

A program for this purpose should be designed exclusively for the mini-computer. Ideas and procedures for such a system are outlined, where the secondary storage access are essentially sequential. For this purpose the data are arranged properly before execution, and points, determined by photogrammetry only, are optionally eliminated before adjustment. The principles are equally applicable on direct-solving methods and on conjugate-gradient methods.

SUMMARY

The adjustment of blocks with both photogrammetric and geodetic observations is a laborious task for any computer. A general data structure is proposed here, which uses terms and definitions from graph theory. This data structure simplifies the manipulation and sorting of the different observations and supplementary information in a way that optimize the data flow through the computer. It also allows different sets of geodetic observations to be included in the structure in the same way as the image observations. In this way, the entire system is optimized and the geodetical observations are treated as part of the system, not as a separate addition. Many other types of observations can also be included in a natural way, e.g. statoscope observations, à priori constraints and photogrammetric model observations.

DATASTRUCTURE

In computer theory, field of interest for the analyze of this problem is graph theory. Graph theory deals with nodes and arcs between the nodes. In block triangulation, the nodes can be projection centres and object points and the arcs thus are the rays of light. The arc between point node A and projection centre node 1 also is the measurement of point A in image 1. This type of graph, where there are two groups of points is a kind of bipartite graph. A feature of this graph is that it has no arcs within the groups, only between them. Now, if you have, say, a distance measured between two object points, this will create an arc between them. If you want to avoid this, because you want to treat all observations as uniformly as possible, you can create a pseudo-node among the image nodes, defining only that a distance between the points are observed. The same applies for a set of directions, which is defined as a pseudo-node with arcs to each point. In this case, the station point must be identified among the target points to facilitate the processing.

The next step is to represent this graph in the computer in a way that avoids the search for data in direct-access data sets. The information needed to build a certain error equation should follow consecutively in a sequential data set. Any supplementary information should also be accesses sequentially, where the latest information, stored in primary storage, can easily be accessed.

A characteristic of the arcs in the above graph is that they represent information, unlike normal arcs. The measurement of point A in image 1 is the arc between A and 1, but it is also a quantitative information. This implies that we should treat the observation as a new class of nodes, each with an arc to an image node (or a pseudo-node) and an arc to an object point node. Now we have three classes of nodes, each with its own information: The target point nodes have their co-ordinates, the projection centre nodes have their orientation parameters and the observation nodes have the image co-ordinates. As we are going to build error equations line by line, observation by observation, it is the observation nodes that are of primary interest.

If we look at an image co-ordinate file, it is normally ordered with all observations in an image kept together, image to image. We can mentally separate the information part (the co-ordinates) from the arcs (we can put image numbers on each co-ordinate record) to get nodes like this

image link	point link	image co-ordinates	other information

Here, a link is a computer representation of the arc, and is simply an address to the image and point respectively. It is also possible to add other information to the node. This node is of fundamental interest in the following discussion, where it is treated in different ways for different needs.

FIRST APPROACH

We start with some presumptions. The blocks have been triangulated preliminary to detect blunders and to yield provisional co-ordinates for object points. Also camera orientation values have been estimated. To build error equations with the observation nodes sorted according to image number (the "natural" order), you access images sequentially and ground point co-ordinates at random. If you sort the nodes according to point links, you access points sequentially and images at random. Since you then treat all measurements of one point at one time, you can eliminate its co-ordinates from the unknowns of the system without further trouble. However, the random image access will cause problems when building normal equations. This is not true when you use the conjugate-gradient solving method, where you do not build normal equations. Instead you store the column indices along with the error matrix elements, row by row.

SECOND APPROACH

In this approach a sorting method is used which gives sequential processing of points and "nearly" sequential processing of images. A little more preprocessing is needed. At first, you have to decide in which direction the block has the largest connection distance. If you minimize bandwidth with cross-strip ordering of the photographs, the largest connection distance are along the strips. Now, you must sort the ground point file to begin in one end of the strips, proceed along the block, and end at the other end of the strips. To do so, you calculate for each point its position along the direction of the strip, that is the quantity $X \cos d + Y \sin d$, where d is the chosen connection direction and X, Y are the (provisional) co-ordinates of the point. This value is stored in the ground co-ordinate nodes. Now the observation node file are sorted according to point number. The two files now have the same order (if not, sort ground co-ordinates files in the same way). Both files are read and the above calculated quantity is transferred to the observation nodes from the ground co-ordinate nodes. Finally, both node files are sorted according to the along-strip position, using the calculated value as key field for the sorting program.

The effect of this preprocessing is that the sequential access of the points will yield a random search of the images, as before. The advantage is that the random search is among only a subset of the images, and not the whole set. It should be possible to keep this subset of image nodes in primary storage, preferrably in a queue, where the least last used node is exchanged when a new image is needed.You will still have problem with the building of normal equations though, and if you do not use the conjugate-gradient method for solving, next method is suggested.

THIRD APPROACH

In the second approach the points were sorted to proceed through the block in sequence. The images linked to the points were accessed in a random way. However, since only a local subset of points were used at a time, only a smaller subset of images needed to be accessed at random. The sorting of points is still not optimal though, no consideration is taken to which images the points are observed in. This can be done in the following way: At first, you must decide the order of the images to minimize bandwidth of the system. The image number should be ordered accordingly, or you can put sequence numbers on the images and keep the other number as a reference only. The second step is to sort the image co-ordinate file (or the observation node file) according to point number. The ground point node file should be sorted in the same way. Now you have all the measurements for one ground point kept together in the observation node file.

If you have done the image ordering well, every point will be measured in images fairly close to each other. You have two choices now, depending on the way you build normal equations. Let us assume that you work with the upper tridiagonal part of them. If you process the normal equations row by row (or rather in sets of rows) you are interested in connections with image numbers higher than the diagonal submatrix. To achieve this sorting, you store in the observation nodes for a ground point the lowest image link number connected to that ground point. You store the same number in the ground node file for the same point. Now you sort the observation nodes primarily according to lowest image link and secondarily according to point link, that is: Within each group of nodes with the same lowest image link, you sort for point link. To be àble to access point nodes easily, you sort them also according to lowest image link. The projection centre nodes are sorted in image order if this was not already done. If you use columnwise processing of normal equations, you must use highest image link numbers instead of lowest, but the algorithm is similar. The same also applies if you use the lower tridiagonal part of the normal equation matrix, but then you must use highest image link for rowwise access and lowest for columnwise access.

TO BUILD NORMAL EQUATIONS

The building, reduction and solving of normal equations is a laborious task for smaller computers. This fact can not be overcome by any sorting procedure. We have in fact created new problems with our sorting schemes. It is now possible for us to reduce normal equations in a simple way, but the access to the rows of the normal equations will be more random, compared to the image by image method. This can, however, be solved by the following procedure.

When we eliminate the ground point co-ordinate unknowns by adding new pseudo or imaginary observations according to Shreiber's method, we get two types of equations. First, we have reduced error equations which hold elements for one image only together with additional parameters elements and right member. Second, we have the added pseudo observations with elements for each image the point is observed in together with elements for the additional parameters and right member. Now, when we build normal equations, the processing of the two types of equation will differ.

For the first type, we can use sequential creation of normal equation additions if we sort these error equations according to image number. This may sound excessive, at first to perform a complicated sorting of the observations, then later to sort corresponding error equations in the original sequence. We must not forget, though, that we have already exploited the advantage of simple reduction. We also keep the sorted observations in proper order for the next iteration. If the computer system does not allow the sort program to be linked to the triangulation system or for other reasons, we can avoid the sorting. In this case, we must look at the way the images are scattered along the error equation file. If we look at one image, its equations can only exist in a limited part of the file. This is because of the presorting of observations.

The next thing to consider is if you use lowest or highest image link number in the presorting. Let us choose lowest link for this example, a similar procedure exist for the other case. Now, you build the diagonal submatrices of the normal equations one image at a time. Start with the first image. When reading through the error equation file, you check for each point its lowest image number. When that number changes, you can stop reading. Also save the address for that equation for further use. Now, you store the produced normal equation submatrices and proceed with the next image. You have to start from the beginning in the error equation file again. When image number 2 is completed, save the address for next equation as above. Continue in the same way. When the image number is greater than the bandwidth, you do not need to search the file from the beginning, but only from "bandwidth" images back. You use the addresses stored earlier for that purpose.

The second type of error equations will create additions to the diagonal submatrices and produce off-diagonal elements in the normal equations. The processing of these equations will not differ significantly from the first type, apart from the fact that you can not sort this file to get sequential access. Now, why do we need to separate these equations in two parts? For one thing, the record size for the two types differ largely. It is hard to administrate direct access files with varying record length. Both files will be stored compactly. A large buffer area for these files can speed transfer significantly.

TO SOLVE THE NORMAL EQUATIONS

The solution of the normal equation is obtained in the usual way. Any band/border equation solving routine with enough capacity will do. If your routine can not handle the border, it is possible to modify it if you can put more than one column on the right hand side. You simply put the border part on the right hand side along with the constant part. When the elimination part is ready, you must continue with the elimination of the border part of the equations. You must then backsubstitute the border part, solve these unknowns and see to that the constant column is modified accordingly. Backsubstitution of the rest of the parameters can then be done as before by the band algorithm, now with only the modified constant column on the right hand side.

POSTPROCESSING, UPDATING OF APPROXIMATIONS

The image node file is simply updated with new orientation parameters. The ground co-ordinate unknowns can be calculated in several ways. One way is to resect each point from the respective images. The data for this are easily obtained from the unreduced error equations which should be solved for the purpose in a separate file. Image parameters and additional parameters are substituted in these equations and they are solved for one point at a time. As the equations are sorted in the same point order as the ground nodes, access to both files can be sequential. The ground co-ordinate parameters could be calculated in other ways, but as the additional parameters affect the solution, the proposed way should not be significantly slower. The question also arises, if we shall modify the image co-ordinates with the additional parameters, if we shall apply the parameters when calculating the right member for next iteration or if it is possible to ignore them completely. Only practical experience can help in this problem. The best solution depends much on the magnitude of the image deformations.

The calculation of residuals in image co-ordinates can be performed parallel to the ground co-ordinate updating. As all observations of a point appear collected, there is a possibility to print out a rms image residual value for each ground point separately, something which might be useful for error detection.

A PRIORI CONSTRAINTS

We have up to now described the way photogrammetric observations can be handled. A special type of node is defined for observations. An important facility in many photogrammetric triangulation systems is the possibility to include pseudo observations for the known (reference) ground co-ordinates, to be able to estimate errors in them. To include this in our system, we must define a special observation node, containing only a link to the reference point. The image link will be empty. This node will fit among the other observation nodes and after the presorting it will appear together with the other observations of the same point. We can process it accordingly. This kind of observation node can be identified by the empty (e.g. zero) image link.

INCORPORATION OF GEODETICAL OBSERVATIONS

In the previous discussion, we have mentioned that geodetic measurements can be incorporated and defined as pseudo-image nodes. Let us use a set of directions as an example. This pseudo node will only hold a link to station point unknowns, an orientation quantity and weight information. The observation nodes will contain the direction, and links to the set-of-directions node and to the target point node. The set-of-directions node is treated as an image node and the target point node is an ordinary ground point node.

To include the set of directions in one equation system we must keep the point co-ordinates involved as unknowns. For each point found that is observed geodetically, we can not eliminate its co-ordinates as unknowns. We also have to form the error equation for the observed direction. To do that, we need the co-ordinates for the other point involved. The other point is always the theodolite station, since we have no observation node for that point.

A link to the station point node is kept in the pseudo node for the set of directions. There we will also find the orientation quantity approximation and weight information. Now, the only problem left is to position the point co-ordinate parameters in the equation system. We can easily spoil our nice band-bordered structure if we have measured directions across a large part of the block. This will create submatrices outside the band which will create problems in the elimination. The remedy for this is to locate the point parameters in the border part. The addresses to the equation elements can be stored in the corresponding point nodes. The additional computer work to solve the border part increases only linearly with border width. This is a more favourable approach than to try to resort the images to yield a minimum bandwidth including geodetic observations. In the border we must also place the orientation parameter unknown. It is not as simple to eliminate this as in geodetic applications, since the directions are processed at different instants.

To avoid an excessive growth of the border, some point parameters can be kept in the band part, namely those who are measured locally in this block. They will create elements within the border which can be kept there. They will, however, destroy the nice six by six submatrix structure of the equations. Links to equation elements have to be stored in the image and point nodes to keep control of the addressing. This will, however, not create any serious problem in the program design, as the nodes are readily available when needed.

The inclusion of point parameters in the border for cross-block measurements will make the border wider and will also introduce many zero elements there. We must be aware of that we should not have a majority of ground points measured in such a way. For normal blocks though, the number of points measured geodetically will be small compared to the total number of ground points, and such a block can easily be adjusted with the outlined method.

ADDITIONAL PARAMETERS

The parameters for image deformations appear in the border part of the solution. It is possible to have a common set of additional parameters for the whole block, or to have different sets for different sets of images. The links to the different parameter sets are stored in the image nodes. For ease of programming, only one type of parameters should do. Also the camera constants can vary, and a link for this purpose should also be stored in the image node. The link will address the camera constant in a small table, kept in primary storage.

CONCLUDING REMARKS

The system outlined above is only existing on a system work level. The central ideas to solve this task are fixed to a great extent, but there are many details of the system which still are to be solved. The data structure proposed is very suited to analyze the system from a theoretical point of view. It also makes the system structured in such a way that you can overlook it easily. One of the great advantages is also that observations of different kinds are treated in the same way. This does not only help in the programming, it simplifies also the inclusion of new types of observations. I have not yet discovered an observation type which does not fit into the system.

For model triangulation, the same data structure can be used. The image type node will be replaced by a model node, containing the seven parameters for orientation of the model and appropriate links to added parameter sets and other information which are relevant for the model. In fact it should not be impossible to include both types of triangulation in the same program. A model can be thought of as another set of observations. I will not go into further detail about this.

I have earlier discussed the conjugate-gradient method for solving this kind of task. For smaller computers this is in my opinion the best way to implement the system. The laborious creation of the normal equations are avoided, and for a limited number of unknowns, the solution will be obtained using only sequential access. As we can eliminate the majority of the ground points very large blocks can be solved this way on a minicomputer. A good review of this solving method is found in Elfwing 1978. There are also several textbooks treating conjugate-gradient methods.

REFERENCES

Bertziss 1975. Datastructures, theory and practice, second edition, Academic Press 1975, 586 pages

Elfwing, T 1978. On computing generalized solutions of sparse linear systems with application to some reconstruction problems. Linköping Studies in Science and Technology. Dissertations No. 27, part A

Elphingstone, G M 1974. Large Block SAPGO Program. Annual Convention of the ASP, St Louis, Mo., March 1974. Also in Phot Eng and Remote Sensing 41(1):101-111, 1975.