# Further Use of Givens Transformations in On-Line Phototriangulation

Knut Ragnar Holm, research scientist
SINTEF / RUNIT – The Computing Centre at the University of Trondheim
Elgeseter gt. 10
N-7034 Trondheim
NORWAY

Commission III

## Abstract

Fast algorithms for Least Squares adjustment and blunder detection are required for the continued development of on-line phototriangulation systems. This applies to industrial as well as to aerial photogrammetry, not least when digital images are involved – and when the process approaches the "near real-time" case. Some comparative tests of different sequential adjustment algorithms have been reported lately. Continued research confirms the impression that an algorithm based on Givens Transformations is faster than other algorithms tested so far. This paper discusses ways of utilizing a Givens Transformation algorithm in on-line phototriangulation systems in order to further exploit its advantageous properties. The basic idea is to divide and distribute time-consuming tasks among the computer's idle times during the coordinate recording procedure.

## 1. Introduction

During the development of On-Line Phototriangulation (OLT) there has been a continuing search for fast algorithms. In recent years progress has been achieved in sequential adjustment algorithms. However, most of the improvements apply to the pure updating of a solution vector within a linearized equation system. Blunder detection procedures still require a substantial amount of time compared to desired response times, and so does a new linearization of the entire equation system after updating the unknown parameters. Based on the latest developments in sequential adjustment algorithms, the author will address these problems, and suggest possible solutions.

The first basic method for sequential updating of a solution vector after adding one or more observation equations to a Least Squares (LS) problem already solved, used the principle of a *stationary Kalman Filtering*. The update was based on updating the inverse of the normal equations matrix (e.g. Mikhail,Helmering,1973; Dowideit,1980). Another principle is to perform the update directly to the factorized normals, and then get the new solution by a back substitution. *Triangular Factor Update* (TFU), as presented for phototriangulation in Gruen,1982, was tested and compared with an algorithm of Kalman form (Wyatt,1982). The results reported show that TFU was faster by a factor of 2–7 for one added object point, and 20–30 for eight added points (Gruen, Wyatt,1983). Also, required data space was far less for TFU.

Use of *orthogonal transformations* belongs to the same category of update methods as TFU, as the update takes place in the factorized normals. Earlier described for general LS adjustment (e.g. Lawson,Hanson,1974; Gentleman,1973), and later proposed for photogrammetry (e.g. Blais,1983; Gruen,1984), an implementation and test of an algorithm based on *Givens Transformations* (GT) was reported in Holm,1986. The test included a comparison with TFU, and indicated that GT would be faster than TFU, at least for few added points. More extensive tests have later confirmed that GT is 2–3 times faster for adding one object point by stereo measurements (two photo points), and 3–5 times faster for deleting one photo point from the equation system (Holm,1988). Compared to TFU, GT incurs less increase in computer time with increasing size of the normals. See Figure 1 (All figures are found behind **References**). Other advantages also apply, e.g. less required data space. An independent test reported in Runge,1987 shows similar results for update times.

The present implementations of both TFU and GT may well be refined for higher speed. However, GT will remain advantageous considering speed (at least for small updates), space requirements and simplicity concerning software implementation and maintenance. If an algorithm with Givens Transformations without square roots is used, an extra improvement of 20% is expected for update time, and back substitution will also become faster (Gentleman,1973).

## 2. Practical value of differences in performance.

Space requirement can be an important factor in choosing an algorithm. According to an example in Holm,1988, the GT algorithm may save 30 Kb compared to TFU in the computer memory in a program capable of handling 9 photos, 175 image points and 45 object points in one block. This may or may not be significant, depending on the computer used and other software residing simultaneously in memory. This paper will concentrate on the computing speed.

Whether the demonstrated advantages of the GT algorithm are significant or not, depends, as far as speed is concerned, on the factors:

A. response time required in the actual application,
B. the amount of computations required within this time (critical operations),
C. the speed of the actual computer.

The values for CPU time used in this paper refer to a specific computer (VAX 11/750). But the relative differences given in % or as a factor, are also applicable on other computers. So far analytical plotters have been equipped with slower computers. A real-time system may have far higher speed. Apart from the absolute times given as examples, the following considerations are general and applicable for fast and slow computers – and for strict as well as relaxed time specifications.

The relations between the factors A, B, and C may be considered in different ways. The common view is to assume that A, B and C represent fixed quantities, and to consider the fulfillment of A (2.1). Another view may be to adjust B to A and C (2.2).

### 2.1. Computer, critical operations, and required response time are fixed.

1] If both algorithms yield response times far above A, it will probably not be sensible to implement either of them.

2] If any of the algorithms gives response around the required response time (A), even small differences may be important as to whether the requirements are met or not.

3] If both algorithms give response times far below A, their relative speed is irrelevant – at least in the first place. However, in this case it may become fruitful to consider another view.

### 2.2. Adjusting amount of work to required response time on a given computer – or Exploiting Fast Algorithms.

Consider the case 3] above, i.e. with a specific computer (C), the critical operations (B) are finished within a fraction of the stated response time required (A). An example of this case is a system for coordinate registration, where B in the first place ("critical operation") consists of receiving and storing image coordinates, and computing and storing model (or object) coordinates, as soon as the operator has pressed the foot switch. The system must be ready to receive the next point whenever the operator presses the footswitch again on the next point. The requirements A should be set according to an assumed *least* amount of time expected between two consecutive point readings, e.g. 10 seconds (The author does not claim this to be a well-founded limit for OLT applications).

1] As a first step some more computing may now be put into the sequence, taking care that A is still not violated. The extra task is assumed to be relevant once for each coordinate registration. And in order to have any sense, it should comprise computing which otherwise has to be done later, at that stage causing an unwanted delay. Section 3.1 gives an example of this procedure.

2] As a further step the *probable* time between coordinate readings may be considered. Even if it is possible to position and read a new point in 10 seconds (and that is what A should reflect), it may in most cases take far more time to move to the area, choose the proper point and position it carefully. According to operators, actual times during aerial triangulation may be 30 seconds or even several minutes. It could be possible to fill this idle time for the computer with some of the work that otherwise will cause an unpleasant delay later. However, in this context the extra work must occur as a number of small operations, which must be handled in such a

way that whenever the operator presses the footswitch, the current additional operation can be finished and the real registration procedure can be completed within the required response time. This is basically the same principle as "time-sharing" in computers. But the solutions proposed in this paper may be implemented in the application programs without any interrupt handling or other computer-specific programming.

If the additional operations make use of the algorithms in question, the choice of algorithm may highly influence the amount of extra work possible during the available time. In both these examples even a difference of 20% in computing speed between two algorithms may be important.

## 3. Suggested procedures for exploiting fast algorithms.

Normally, a reduction of 80% in computing time for a task which uses only 10% of the total time in a specific sequence will have practically no value for improvement of the response time for that sequence (though, it would be natural to choose the "best" alternative, even if the differences are small). However, if, by utilizing the fastest algorithm, one could move 90% of the computing to preceding sequences, this would give significant improvement. The following sections show ways of utilizing a fast algorithm for sequential updating of Least Squares adjustments along the lines mentioned above. Only the first procedure (3.1) has been implemented and used by the author. The two others are just outlined, and have not been tested.

### 3.1. Update the factorized normals within the point reading sequence.

Figure 2 shows CPU time used for back substitution in normal equations of various sizes. Some values for the updating are also shown. The values used as examples refer to a photogrammetric blocks of 6 to 9 photos, with maximum size of the altered part of the normals (i.e. worst case).

According to Gruen,1982 a natural procedure in on-line triangulation is to read points in groups of, for instance, 6 and then do the update, back substitution, and data snooping. Thus the operator does not have to wait for the completion of a time consuming process after each reading. Leaving out the blunder detection for a while (and recalling that the numbers are *examples* from a VAX 11/750), Figure 2 shows that the update with 6 stereo-measured points by TFU takes about 1–1.8 seconds (s), while the back substitution for all unknown parameters requires 0.4–0.6 s. If the update may be avoided at this stage, the solution vector would be available after 0.4-0.6 s instead of after 1.4–2.4 s, which could be a significant improvement. Using GT would give almost the same relations. The photo parameters are available within a fraction of this time (<0.05 s), so the improvement is far better if not all the object point's coordinates are required.

This improvement can be achieved by updating the factorized normals after each point reading (see 1] in 2.2), which is possible if the update time for one point reading may be added to the sequence without violating the response time requirement. According to Figure 2, GT will use 0.2-0.5 s while TFU uses 0.5-1.3 s for this update. Both will probably do in OLT. However, the factor 2.5 between the two may become significant on a less powerful computer, or if it is desirable to put more of the computing in beween the point readings.

### 3.2. New linearization of the equation system by continuously exchanging observation equations

A difficult problem in fast sequential least squares adjustment is caused by the time required to update the normals after re-linearizing the observation equations with updated parameter values. This section presents some ideas as to how a fast updating algorithm may be used to solve this problem.

#### 3.2.1. The problem

The LS adjustment is performed on a linear equation system. This applies to sequential as well as simultaneous solutions. When the functional model is non-linear, as is the case in bundle adjustment, the equations have to be linearized. A general formulation of the functional model may be

$$E(\mathbf{l}) = F(\mathbf{x}) \qquad (1)$$

E(**l**) represent the expectation of the observations **l** and **x** contains the unknown parameters. Linearized about an approximate value $x_0$, (1) yields

$$E(\mathbf{l}) = (\partial F(\mathbf{x})/\partial \mathbf{x})_0 d\mathbf{x} + (F(\mathbf{x}))_0 + \Delta \quad (2)$$

where $\partial F(\mathbf{x})/\partial \mathbf{x}$ is the Jacobian matrix of $F(\mathbf{x})$, the subscript $_0$ indicates that the functions are evaluated in the value $x_0$, and $\Delta$ contains the elements of higher order. Calling the evaluated functions **A** and **f** respectively, and applying actual measurements **l** and true errors -**v**, the error equations are obtained as

$$\mathbf{l} + \mathbf{v} = \mathbf{A}_0 \, d\mathbf{x} + \mathbf{f}_0 + \Delta \quad (3)$$

$\Delta$ will be small when **dx** is small. With the Gauss-Newton method the approximated error equations (4a) with $\Delta$ omitted, are solved successively by LS, as the unknown parameters each time are updated according to (4b):

$$\mathbf{v} = \mathbf{A}_i \, d\mathbf{x}_{i+1} - (\mathbf{l} - \mathbf{f}_i) \quad (4a)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + d\mathbf{x}_{i+1} \quad (4b)$$

Assuming equal weight for all observations, and introducing $\mathbf{d}_i = (\mathbf{l} - \mathbf{f}_i)$, the normal equations may be computed as

$$\mathbf{A}_i^T \mathbf{A}_i \, d\mathbf{x}_{i+1} = \mathbf{A}_i^T \mathbf{d}_i \quad (5)$$

which after factorization ends up as

$$\mathbf{U}_i \, d\mathbf{x}_{i+1} = \mathbf{h}_i \quad (6)$$

When performing sequential estimation by Givens Transformations or Triangular Factor Update, the updates are supposed to take place more or less directly in (6). The reason for a sequential adjustment process is that the result should be available within shorter time than what is possible for the simultaneous solution. A complete exchange of all re-linearized observation equations is therefore not desirable after each update, since that is effectively nothing less than an iteration of simultaneous adjustment.

In Mikhail, Helmering,1973 this problem is addressed. It is suggested that the new observation equations may be linearized in the last version of the parameter vector, without further changes in the normals. The result will not coincide with a simultaneous solution, because the observation equations are not linearized in the same values. A test indicates that the solution will become satisfactory, with reservations that the test was rather simple. Besides, the conditions were different from those occurring during OLT.

Gruen,1984 states that updating the parameter vector should not be allowed without also updating all observation equations, i.e. re-linearizing them. Otherwise the equation system becomes inconsistent, and parameter solution as well as error detection will not be reliable. A couple of procedures for tackling the problem are mentioned. One is simply to provide sufficiently good approximate values that it is unnecessary to update them at all during the triangulation. Even though it might be possible to compute such values in aerial triangulation, it appears too risky to base an OLT system on this assumption. For industrial photogrammetry it is not applicable, except when using fixed camera stations. And the method excludes the possibility of getting the final solution from the on-line procedure.

The other solution mentioned is to perform a complete simultaneous solution from time to time. This is rather time consuming compared to the sequential procedure, and hence should preferably be done at times when the operator is busy with other tasks. This might be during exchange of photos, which may give some frustration if it results in detecting a blunder after removal of the actual photo. The most optimal time concerning reliability would be as soon as possible after introduction of a new photo, since that is the time when the largest corrections are likely to occur. On the other hand, this may cause undesirable delays for the operator.

Veress,Huang,1987 gives a method for sequential estimation where new equations may be linearized in the updated parameter vector without the drawbacks mentioned in Gruen,1984. The solution is a Kalman-type algorithm where corrections after each sequence are made to the cofactor matrix $Q_{xx}$ as well as to the parameter vector **x** itself. However, it appears that the following formula is a part of the procedure:

$$Q_{xx1(new)} = (A_{1(new)}^T A_{1(new)})^{-1}$$

where $\mathbf{A}_{1(new)}$ represents the old observation equations re-linearized in the updated parameter vector. In other words, this procedure, too, implies a complete new simultaneous adjustment with the updated parameters as the linearizing point. Consequently the method is not convenient when speed is the reason for applying sequential adjustment.

### 3.2.2. A suggested procedure.

Assume a consistent normal equation system (5), factorized to yield (6). As long as the same parameter vector $\mathbf{x}_i$ is used for linearization, additional observation equations may be added to (6) by TFU or GT updates, implicitly causing $\mathbf{A}_i$ and $\mathbf{d}_i$ to grow larger – by adding rows. When the solution is computed and the parameter vector is updated (4b), the entire equation system is to be updated as well. A new iteration of simultaneous adjustment would now consist of deleting the entire equation system and building and factorizing new normals based on $\mathbf{A}_{i+1}$ and $\mathbf{d}_{i+1}$, now linearized in the new parameter vector. The following alternative procedure might be applicable, provided that the operator does not necessarily need the solution of the updated equation system before continuing the point readings.

The suggested procedure depends on a fast updating algorithm, and consists of sequentially exchanging the observation equations instead of deleting and re-building the entire equation system. For each observation equation, the old version is removed sequentially and the updated version is immediately added. According to the principle of 2] in 2.2, this may be done as long as the computer is idle between the point readings. As soon as a new reading is made, the actions belonging to the point reading must be performed, and then the sequential exchange of equations may continue. When all equations are exchanged, the solution vector may be computed, the parameter vector updated, and the exchange procedure may start again. Figure 3 indicates how the content gradually changes in the coefficient matrix (**A**) which is the basis for the current factorized normals.

Using times from the test reported in Holm,1988, and assuming that the factorized normals are always updated immediately according to the new point reading (3.1), there may still be time for the exchange of 5–50 observations within a limit of 5 seconds (using GT). As mentioned before, there will most often be more time available, and thus a larger amount of observations may be exchanged.

The procedure outlined is fully equivalent to a new simultaneous adjustment iteration, but it has the disadvantage that new solutions as well as blunder detections may occur only at the stages where all the observation equations are of the same version. The complete exchange of equations may last over several point readings. If the operator wants to see the updated results, including blunder detection, in the middle of an exchange sequence, he will have to wait until it is completed. Another possibility is to modify the procedure in such a way that the factorized normals are consistent – at least within acceptable limits – at each point reading.

### 3.2.3. Alternative procedures

Newton's method for solving a non-linear equation system $G(\mathbf{x}) = 0$ may be expressed as

$$\mathbf{x}_{i+1} = \mathbf{x}_i - G'(\mathbf{x})_i^{-1}\, G(\mathbf{x})_i \qquad (7)$$

where $G'(\mathbf{x})$ is the derivative of $G(\mathbf{x})$, i.e. the Jacobian matrix. A modification called "quasi Newton" is

$$\mathbf{x}_{i+1} = \mathbf{x}_i - G'(\mathbf{x})_0^{-1}\, G(\mathbf{x})_i \qquad (8)$$

implying that the same Jacobian is applied in all iterations. Both methods may converge, but (8) has only linear convergence while (7) has quadratic convergence (Johnson,Riess,1982 p 201). "Quasi Newton" is also found in modified forms, like for instance evaluating $G'(\mathbf{x})$ at some stages during the iterations, giving $G'(\mathbf{x})_j$, thus achieving faster convergence. Obviously, at a certain iteration cycle

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{B}^{-1}\, G(\mathbf{x})_i \qquad (8)$$

the new value $\mathbf{x}_{i+1}$ will become a better approximation the closer $\mathbf{B}$ is to $G'(\mathbf{x})_i$. Consequently, starting with $\mathbf{B}=G'(\mathbf{x})_0$, a better result would be achieved if one or more rows of $\mathbf{B}$ was exchanged with rows from $G'(\mathbf{x})_i$. In the Least Squares case, $d\mathbf{x}_{i+1}$ corresponds to $-G'(\mathbf{x})_i^{-1}G(\mathbf{x})_i$, where $G'(\mathbf{x})_i^{-1}$ may be regarded as the pseudo inverse of $\mathbf{A}_i$, i.e. $\mathbf{B}_i^{-1} = (\mathbf{A}_i^T\mathbf{A}_i)^{-1}\mathbf{A}_i^T$, and $G(\mathbf{x})_i$ corresponds to $\mathbf{d}_i$.

With this background, the procedure after update of the parameter vector may be modified in different ways. In addition to two procedures directly referring to "quasi" and "modified quasi" Newton, an additional procedure is suggested for certain stages of the triangulation.

> Note that neither of these procedures conforms to the case of different generations of observation equations in the equation system. The problem with the latter is best demonstrated by an example: Suppose the equation system was solved and the parameter vector updated by adding $d\mathbf{x}_i$. Since then a few equations have been added (or exchanged) with the updated linearization point, while most of the equations are unchanged. Then a new solution $d\mathbf{x}_{i+1}$ is computed. Since most of the equations are exactly as before, $d\mathbf{x}_{i+1} \approx d\mathbf{x}_i$, and the question is what version of the parameter vector should be updated. If the last version is updated, it implies applying practically the same correction twice. No wonder if the parameter vector in this case "drifts off" after a number of updates (Gruen,1984). On the other hand it is of no use to update the previous parameter version, since the already applied $d\mathbf{x}_i$ must be regarded as the best correction at that stage. The main difference in the methods is that the quasi Newton starts with updating the *entire* right-hand side of the equation system.

*a]* *Re-compute (re-evaluate) the entire constant vector.*

Conforming with the quasi Newton method, after updating the parameter vector a new right-hand side of (5) is computed and factorized, based on the re-evaluated constant vector $\mathbf{d}_{i+1}$ – but on the old coefficient matrix $\mathbf{A}_i$ of the observation equations (4a). Introducing new observation equations linearized completely in the last parameter vector will imply the introduction of a modified quasi Newton method.

Data snooping will be more reliable after updating the constant vector this way, because the residuals, as well as the parameters, will better approximate the final estimate. A complete re-linearization will give a larger change in the constant vector, and hence in the residuals, than in the coefficient matrix **A**, and hence in the $\mathbf{Q}_{vv}$ matrix. Consequently, leaving **A** unchanged while updating **d** will have minor impact on the reliability.

Normally the CPU time required for re-linearizing and updating the right-hand side is much less than updating the complete equation system. If it appears difficult to keep within the prescribed response time, it may be necessary to divide this task also, allowing updates to occur more quickly at new point readings. But if a blunder detection is requested it will probably not appear too inconvenient to wait for completion first.

*b]* *Sequential exchange of rows in the coefficient matrix only.*

In addition to the update of the constant vector, a sequential exchange of the rows in the coefficient matrix **A** will improve the convergence, and the reliability of the data snooping as well. Conforming with the "modified quasi Newton" methods, this procedure will only be valid after the constant vector update is completed. Hence it must be regarded as an extension of the procedure *a]*.

*c]* *Exchange of selected observation equations.*

When new photos are introduced in the block, the first approximations for their orientation parameters are far less accurate than the parameters already adjusted in the block. Therefore the new parameters will also get the largest corrections in the first few iterations. A convenient procedure for the first few iterations after introducing a new photo could then be to compute the corrections and update the new photo parameters only. Then only those observation equations containing the updated parameters are exchanged in the factorized normals as in section 3.2.2, i.e. completely. Looking at the algebra, it appears that the equation system would remain consistent in this case, because as no other parameters are altered, any attempt to "re"-linearize the rest of the observation equations would result in no change at all.

When new object points are added, their coordinates should also be included in the parameter update. In order to maintain consistency, the re-linearization and exchange of observation equations must then include all observations of these points, also in old photos.

### 3.2.4. "Cleaning up".

Long series of sequential updating and "downdating" according to these procedures, *may* lead to some distortion of the equation system, due to propagation of rounding-off errors in the computer. Although it is not expected to cause severe errors – at least not when utilizing the numerically stable GT algorithm – a complete simultaneous adjustment should be accomplished at certain stages. As this task has the purpose of "cleaning up", or "consolidating", and no significant changes are expected, it may well be carried out at times when no measuring is going on. In this case a suitable time would be during exchange of photos in the instrument. Then there will be time for a sufficient number of iterations, as well as a complete data snooping, so that the block will be optimally adjusted and ready for the continued triangulation with the next photo.

### 3.3. Continuous updating of constant vectors for the Unit Vector Method of datasnooping

Error detection by data snooping is the most time-consuming task during on-line triangulation. Therefore improvements to reduce the delays caused by data snooping are highly desireable. In Gruen,1982 it is suggested that, in order to prevent the delay between each point reading, the data snooping should take place only after a number of readings. Further, a method for data snooping is described which makes it possible to do the test for a selection of the observations, thereby saving computer time compared to the full data snooping. The method is based on solving the normal equation system with a new constant vector for each of the observations to be tested. Each constant vector is based on the replacement of the constant vector of the observation equation system with a unit vector, where the 1 corresponds to the observation to be tested. Therefore it is called the "unit vector method" (For further description, see Gruen,1982).

The bulk of the computing time in the unit vector method is spent building and factorizing the constant vectors of the normals, and the back substitution. Referring to results from the same test as before, data snooping of 170 photo points in a block with 9 photos takes 25 s. If building and factorizing could be performed at earlier stages, i.e. during the update after each point reading, the rest of the task could probably be done in 10–15 s. Testing a subset of 50 photo points would accordingly take 3–4 s instead of 7 s.

A possible way of achieving this is to keep and maintain all the factorized extra constant vectors along with the original equation system, as indicated by the matrix **H** in Figure 4. The method requires a lot of extra storage, and the update time may increase by a factor of up to ten (i.e. up to 5 s in the example), depending on the structure of **H**. However, storage requirement as well as time consumption may be reduced if some observations could be left out as "sufficiently tested" during the process. Also, the increase in update time will be far less when utilizing parallel processors, because the only difference from the normals update is that each row (vector) processed by Givens Transformations is extended.

## 4. Conclusions

The suggested procedures have in common the attempt to distribute time-consuming tasks to the times between the point readings, when the computer is idle. The procedures may give more or less significant improvements, depending on computer speed and response time requirements as well as on refinements of the algorithms. The procedures are based on the specific working scheme of the GT algorithm. Although the first two procedures might also be implemented with TFU, this would not give similar benefits. The methods in 3.2 and 3.3 have not been implemented or tested. To get a reliable impression of their performance, practical tests should be carried out.

The first procedure (3.1) is in fact the most natural way of implementing the GT algorithm, since this algorithm basically processes the observation equations one by one – whether there are few or many in an update. It has been successfully implemented and used in the test cited.

So far it seems that a procedure along the lines of 3.2 will give significant improvements under circumstances like those mentioned. The major benefits are better reliability caused by several iteration cycles performed during the on-line triangulation without delaying the operator´s work.

The last procedure (3.3) is just outlined. The improvement indicated may not seem impressive – reducing one sequence from 7 to 3 s by increasing several others from 0.5 to 1–5 s. But it may be relevant in certain circumstances, e.g. if a halving of the CPU time for the error detection is mandatory to give the required response.

## References


**Blais J. A. R., 1983:** Linear Least Squares Computations Using Givens Transformations. Canadian Surveyor 37(4):225–233.

**Dowideit G., 1980:** An On-Line Bundle Block Adjustment for Analytical Plotter. ISPRS 23(3):168–177.

**Gentleman W. M., 1973:** Least Squares Computations by Givens Transformations Without Square Roots. Journal of the Institute of Mathematical Applications, Vol. 12 (1973) pp 329–336.

**Gruen A., 1982:** An Optimum Algorithm for On-Line Triangulation. ISPRS Commision III Symposium, Helsinki 1982.

**Gruen A., 1984:** Algorithmic Aspects in On-Line Triangulation. ISPRS Congress, Commision III, Rio de Janeiro 1984.

**Gruen A., Wyatt A., 1983:** Algorithmic Problems in On-Line Bundle Triangulation. ACSM-ASP Convention, Washington D. C., March 1983.

**Holm K. R., 1986:** Test of Algorithms for Sequential Adjustment. ISPRS Commission III Symposium, Rovaniemi, Finland, August 1986.

**Holm K. R., 1988:** Givens Transformations in On-Line Phototriangulation. Dissertation (in Norwegian), Department of Geodesy and Photogrammetry, Norwegian Institute of Technology, Trondheim, Norway

**Johnson L. W., Riess R. D., 1982:** Numerical Analysis. Addison-Wesley Publishing Company.

**Lawson C. L., Hanson R. J., 1974:** Solving Least Squares Problems. Prentice-Hall, Englewood Cliffs, New Jersey.

**Mikhail E. M., Helmering R. J., 1973:** Recursive methods in Photogrammetric Data Reduction. Photogrammetric Engineering 39(9):983–989.

**Runge A., 1987:** The Use of Givens Transformations in On-Line Phototriangulation. ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data, Interlaken, Switzerland, June 1987.

**Veress S. A., Huang Y., 1987:** A Method for Improving the Efficiency of the Sequential Estimation Procedure in Photogrammetry. Photogrammetric Engineering and Remote Sensing 53(6):613–616.

**Wyatt A. H., 1982:** On-Line Photogrammetric Triangulation – An Algorithmic Approach. Master Thesis, Department of Geodetic Science and Surveying, The Ohio State University.

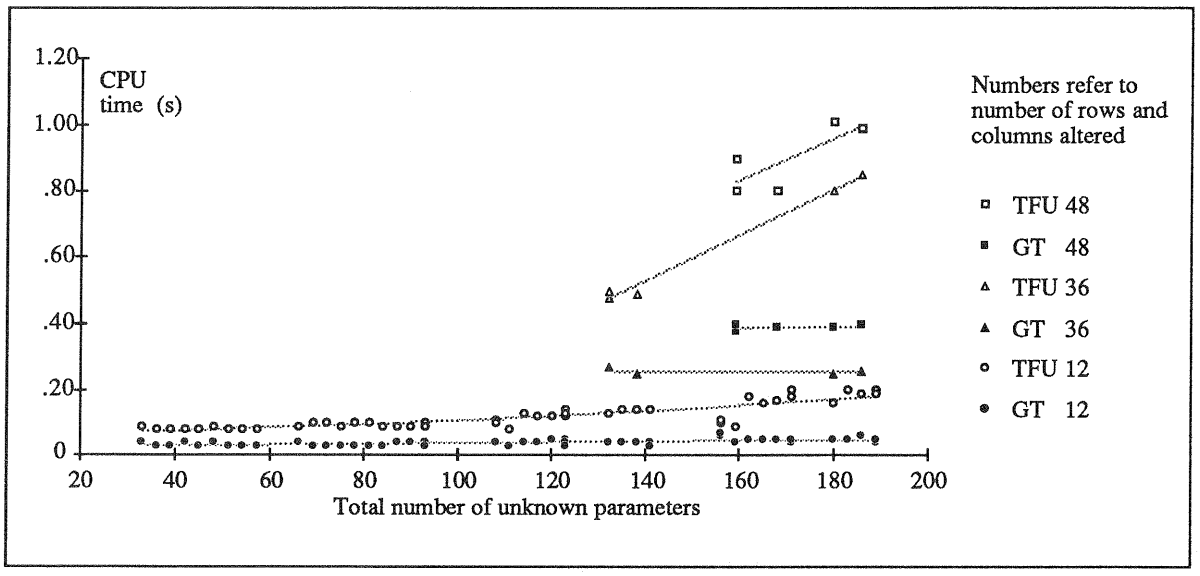


**Figure 1:** CPU-time for updating the factorized mormal equation system with 1 point reading (stereo) as a function of the total number of unknown parameters. The time also depends on the size of the actually updated part of the normals. Note the different gradients for TFU and GT.

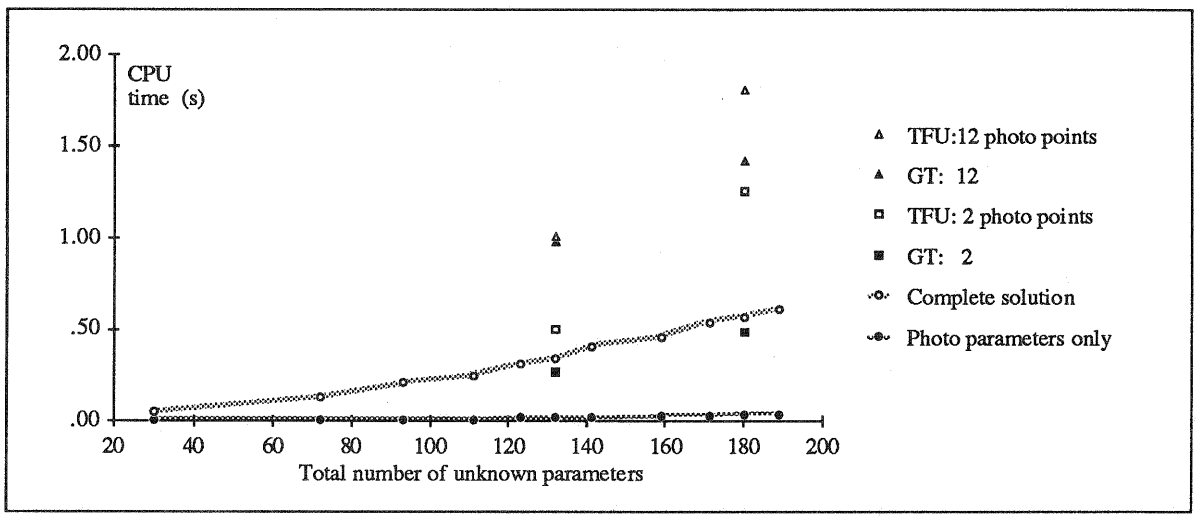


**Figure 2:** CPU time required for back substitution – totally and for photo parameters only – and update times for one and six point readings (stereo). Update times refer to the "worst cases", i.e. maximum number of rows and columns altered and thus highest time consumption, also coinciding with the least differences between the time requirements of the two algorithms.

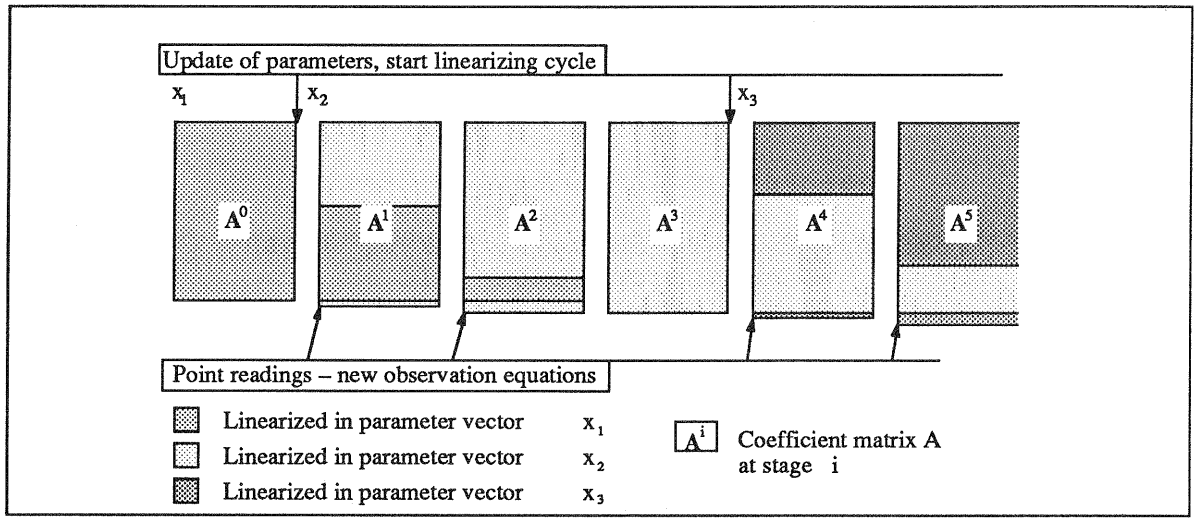


**Figure 3:** Content of the coefficient matrix of observation equations during sequential exchange of observations. New equations, i.e. from new point readings, are linearized in the new current parameter vector.

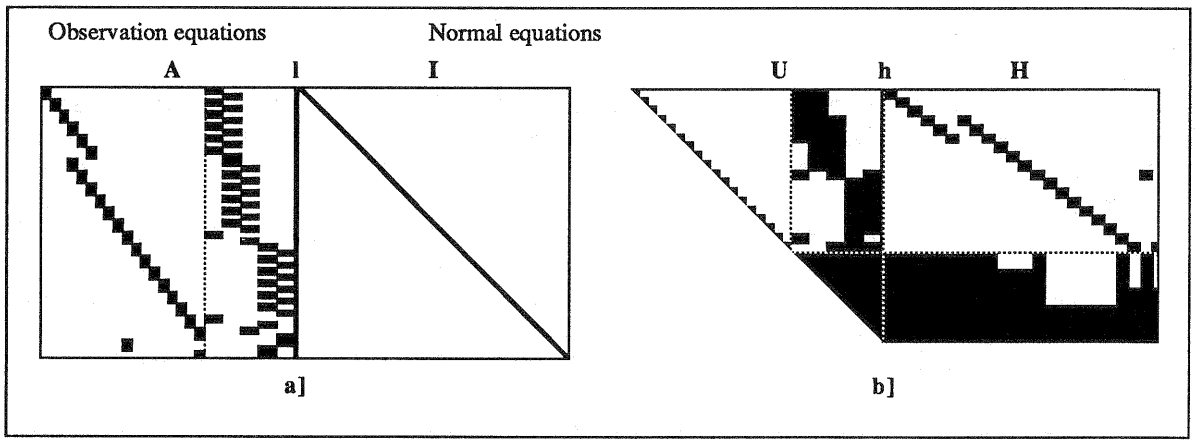


**Figure 4**: Structure of the equation system with the extra constant vectors for the Unit Vector Method.
**a]**: Observation equations, with the extra unit vectors forming an identity matrix.
**b]**: Factorized normal equations with the extra constant vectors forming the matrix **H**.