An Evaluation of Three M-Processor Designs for a Digital Photogrammetric System

Abstract
The common transformation sequence adopted for obtaining digital object space coordinates in almost all modern analytical systems, whether analog-image or digital-image type, is the so called image-model-object space arrangement. In this approach, an image data processor (I-processor) composed of correlators (sometimes involving a human operator) converts the raw image data into feature-based image vector data. Taking corresponding pairs at a time, a model processor (M-processor) converts the image vector data to the model space equivalent, while an object processor (O-processor) reduces the model vector data into object space vector data; all the processors operate in real time. Through a reversion of their operations, these processors can reduce a given object space vector into two image space vectors. Given that the errors and efficiency of other components are fixed, the accuracy and efficiency of the data reduction system is directly influenced by the accuracy and efficiency of the M-processor included in its design. In this paper, three algorithmic designs are considered for the M-processor. These include image space intersection, model-space parallax vector bisection, and model-space y-parallax averaging. These three designs are included in three prototype implementations of a digital data reduction system. Empirical studies involving both real and simulated data show that the processor based on parallax vector bisection in the model space is the most suitable for implementation in a PC-based workstation when cycle time is critical.

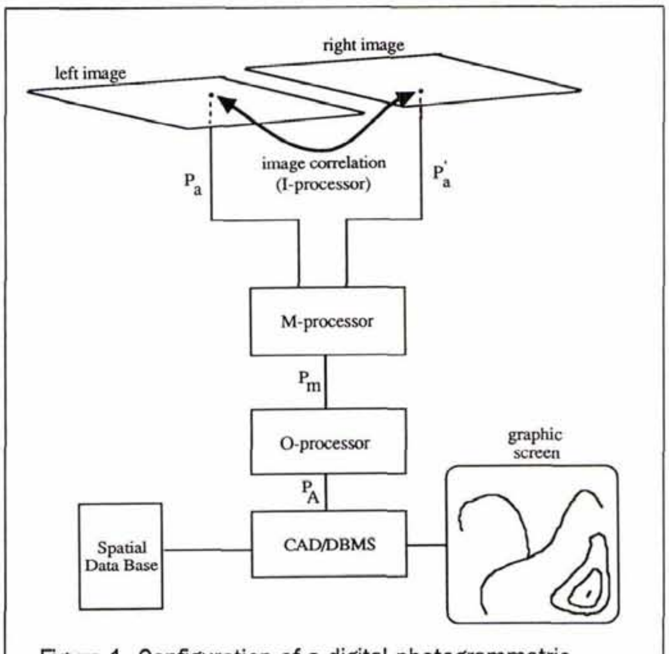

Figure 1. Configuration of a digital photogrammetric workstation.

Introduction
Improvement in microelectronics and the growth of software technology have provided the impetus for PC-based implementations of analytical data reduction systems. Nowadays, due to advances in CCD-scanning technology and the reduced cost and increased capabilities of modern computing devices, low-cost digital data reduction workstations are evolving. Because of their potential for automation of the reduction process, they are attracting the attention of photogrammetrists. Operationally, such a system is an input-output device which converts raw digital image data into the required spatial information in or near real time (Figure 1), sometimes involving the human operator. When supported by a graphics program and a database management program (preferably a GIS), the possible applications of such a system are numerous. These include (1) production of orthoimages (digital orthophotos), (2) 3D map updating, (3) compilation of land-use and topographical maps, (4) acquisition of DTM, etc. For all these uses, raw digital images, obtained from scanned analog images or acquired directly using digital sensors, may be employed (Miller *et al.*, 1992; Schenk and Toth, 1992).

In general, three data processors are involved in the real time operation of such a system (Figure 1), namely, image space processor (I-Processor), model space processor (M-processor), and object space processor (O-processor).

The I-processor, which may include the human operator, is an image space pixel-vector and vector-pixel converter which performs a forward operation involving the measurement or extraction of coordinates of points, lines, and polygonal features from digital images using procedures, such as image correlation techniques (image matching) (Ackermann, 1984) or edge extraction (Hellwich and Faig, 1992). Further-

J. Olaleye*
W. Faig
Department of Surveying Engineering,
University of New Brunswick,
P. O. Box 4400, Fredericton, NB E3B 5A3, Canada.

*Presently with the Department of Surveying, Faculty of Engineering, University of Lagos, Lagos, Nigeria.

Photogrammetric Engineering & Remote Sensing,
Vol. 59, No. 12, December 1993, pp. 1729–1735.

0099-1112/93/5912–1729$03.00/0

more, this processor performs a reverse operation (vector to pixel conversion) in which image space vector data is transformed back to the original digital image. The reverse operation is useful for applications requiring the superposition of vector data on the raster image, such as (1) image space map updating operation for which an existing digital map has to be superimposed on the image, (2) checking the spatial fidelity of digitized features through highlighting of the pixels corresponding to the features in the original digital image as they are being digitized, and (3) real time movement of the measuring marks to certain image locations specified in the object space. In these cases, the I-processor reduces the image space vector data to the appropriate pixel location in the digital image.

The M-processor (mainly a software engine) performs a forward data conversion operation which involves the computation of the model coordinates from two corresponding image space vectors. It employs the principles of analytical geometry to construct mathematically the projective relationship between the image (or measurement) space and the stereomodel space to achieve the data transformation. Moreover, it performs a reverse function in which a given model space vector is projected into its constituent image vector through an inversion of the forward vector operator (Faig, 1984; Ghosh, 1988). This enables the recovery of the corresponding image space vectors that were transformed to the model space as required in applications such as those stated earlier.

The O-processor is a software engine whose forward operation turns a model vector into the object space. It uses the concepts of projectivity between the model space and the object space to achieve the transformation. Through an inversion process, this processor also converts a given object space vector into its model space equivalent.

Technically, the I-processor is very important; and, in practical terms, it is often the most expensive and difficult to implement of all the components of a digital system. The extraction of spatial entities from a digital image inherently is a knowledge-based operation which requires the skill and intelligence of a human operator. But while research is still being conducted towards the development of the so called image expert systems which expectedly will emulate human intelligence and automate the process, many of the current implementations rely, to varying degrees, on the human operator for successful operation.

Nonetheless, the concern of this paper is neither the composition of the I-processor nor of the O-processor, but, rather, the design and operational issues of the M-processor. Consequently, to narrow the discussion to the issues of interest, we require the following assumptions. First, we assume that we have in place an I-processor (of suitable design) capable of supplying the image vector data at a certain optimum level of system efficiency, operational speed, and processing accuracy. Second, because the O-processor invariably uses a standard 3D data conversion algorithm in its operations, it is indeed a fixed system component performing at the maximum possible efficiency. Third, we assume that the available relative and absolute orientation parameters guarantee optimum accuracy of the real time data processing, even though this accuracy is limited in practice by the accuracy of these orientation parameters, which are often obtained by measurement and computations off-line. Therefore, the I-processor, the O-processor, the operating parameters, and the possible effects of these entities on system performance are assumed fixed and are not discussed any further.

Given these assumptions, we focus on the M-processor and note for a start that, because of the amount of computation it has to perform in real time and the many possible algorithms that may be selected for its implementation, it becomes the critical system component whose algorithmic design determines whether or not the optimum system performance will be achieved.

Theoretically, a number of formulations exist for the operation of the M-processor depending on the level of rigor required. These include (1) image space intersection, (2) parallax vector bisection, and (3) *y*-parallax averaging, which are based on the concepts of spatial intersection of corresponding rays. These algorithms have been discussed by several authors (Schut, 1973; Miles, 1968; Ghosh, 1988), some of whom have concluded that, while the image space intersection with rigorous least-squares solution may be considered to give the theoretically best estimates of the coordinates of the point in the model space, in practical terms the differences between the three methods are not significant and they all lie within the precision of the observations from which the model is formed (Schut, 1973; Miles, 1968). Several questions arise, however, when the M-processor is viewed within the context of modern digital systems. Being a critical component of the digital workstation, and performing real time round-loop computations required to transform data from the image space to the model space and vice versa, it is only natural to inquire about the design that provides the optimum system in terms of round-loop accuracy, speed of operation, and overall system throughput. These issues are critical to the performance of the system, particularly in a PC-based implementation.

Because it is a link between the image space and the object space, it is not too difficult to appreciate its influence on the performance of the entire system. In terms of real time operation, the M-processor has a significant influence on the speed of operation of the system, and, quite clearly, this influence can be seen to depend on the complexity or rigor of the chosen algorithm for its implementation. Furthermore, considering the assumptions made earlier, the M-processor determines how much of the attainable accuracy is actually realized by the system; again, this depends on the numerical sophistication of the selected algorithm for both its forward and reverse conversion operations. This paper attempts to answer these questions from a practical point of view. Although the theoretical bases of these algorithms are well known, this paper employs a vector based methodology to present the algorithms. Based on our experience, the vector space approach has more than a purely intellectual value. Besides providing insight into the operational mechanisms of the M-processor and, indeed, of the entire digital data reduction system, it also has a stimulating effect in that it leads to a set of generalized computational schemes which make the implementation of a digital photogrammetric system simple and transparent. Instead of presenting rigorous derivations, we have adopted the option of itemizing the steps involved in the algorithms, which we believe is more useful from a practical point of view. We have included these algorithms in three implementations under VMS on the Micro-VAX-II computer and compared them in order to determine the achievable accuracy and to get an idea of the computational speed for each design. Thus, in the rest of this paper we present the summary of these algorithms and describe the experimental investigation conducted together with the conclusions reached.

Theory of the Vector-Based Algorithms for the M-Processor

Referring to Figure 2, we see that three vector spaces are involved in the operation of the M-processor, i.e., the left image space, the right image space and the model space. A pair

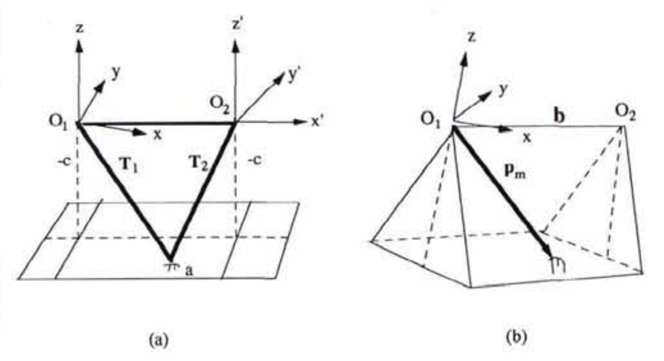

Figure 2. Model vector space. (a) Two corresponding image rays $\mathbf{T}_1$ and $\mathbf{T}_2$, originating at O_1 and O_2, respectively, intersect at model point *a*. (b) The model space is represented by the model vectors $\mathbf{P}_m$.

of conjugate image vector data (Figure 2a), are numerically converted to model space vector data, and conversely, given model space vector data (Figure 2b), are resolved into two conjugate image space vector data. Using the Apparent and Real Directions of Vector Spaces (ARDOVS) methodology (Olaleye, 1992), the vector-based algorithms which achieve these data conversions are easily derived. Briefly, ARDOVS states that, to every photogrammetric vector space, there is a set of natural directions (as seen by elements within the space) and a set of apparent directions (as seen by elements outside the space). When two such spaces are to exchange elements, one space is always the fixed space, which in the ARDOVS concept is called the **R**-space with natural directions **i**, **j**, **k**, and apparent directions $\mathbf{R}_1$, $\mathbf{R}_2$, $\mathbf{R}_3$. The other space is then the movable space with natural directions **i'**, **j'**, **k'**, and apparent directions $\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{C}_3$.

In order to utilize the ARDOVS concept, a decision has to be made as to which space is the fixed and which is the movable. Through an intuitive process, we know that the model space is the fixed space, i.e., the **R**-space, while the two images are movable or **C**-spaces. Consequently, we have a choice of formulating the M-processor algorithm in the image space, in which case, we have to project an element of the **R**-space (model space) into each of the **C**-spaces (left and right image spaces) and require their intersection in the image space. Alternatively, we could formulate the algorithm in the model space, in which case corresponding image space elements must be transformed to the model space. The first of our algorithms uses the image space formulation and, thus, is called image space intersection while the other two algorithms use the model space formulation and are called model space parallax vector bisection and model space *y*-parallax averaging, respectively. In what follows, the computational steps of these algorithms are given. Readers interested in the theoretical developments are referred to the study by Olaleye (1992).

Algorithm Based on Image Space Intersection

In the ARDOVS concept, the functional model which relates any 3D Cartesian space (**R**-space), such as model or object vector space, to an image space (**C**-space) is given as

$$\begin{aligned} k_x\mathbf{C}_3\cdot\mathbf{P}_A + \mathbf{C}_1\cdot\mathbf{P}_A - k_x\mathbf{C}_3\cdot\mathbf{P}_o - \mathbf{C}_1\cdot\mathbf{P}_o = 0 \\ k_y\mathbf{C}_3\cdot\mathbf{P}_A + \mathbf{C}_2\cdot\mathbf{P}_A - k_y\mathbf{C}_3\cdot\mathbf{P}_o - \mathbf{C}_2\cdot\mathbf{P}_o = 0 \end{aligned} \quad (1)$$

where $k_x = x_a/c$ and $k_y = y_a/c$;

$\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{C}_3$ are the apparent direction vectors of the image space (which are similar to the column vectors of the familiar rotation matrix in analytical geometry);
$\mathbf{P}_A$ is the position vector of a point in the 3D space;
$\mathbf{P}_o$ is the vector which locates the origin of the movable space in the fixed space; and
the dots (·) signify inner products of vectors.

In order to obtain the desired vector $\mathbf{P}_m$, Equations 1 are applied to both the left and right image spaces, and a pair of collinearity equations each may be derived as follows:

Left image:
Let $\mathbf{P}_A = \mathbf{P}_m$, $\mathbf{P}_o = 0$, $\mathbf{C}_1 = \mathbf{i}$, $\mathbf{C}_2 = \mathbf{j}$, $\mathbf{C}_3 = \mathbf{k}$ (left image of a dependent pair relative orientation). Putting these in Equation 1, we obtain the first two equations in Equation 2.

Right image:
Let $\mathbf{P}_A = \mathbf{P}_m$, $\mathbf{P}_o = \mathbf{b}$, and evaluate $\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{C}_3$ from the relative orientation elements (right image of a dependent pair relative orientation). Substituting these in Equation 1, we obtain the last two equations in Equation 2.

Thus, two sets of collinearity equations are obtained which, when treated simultaneously in a least-squares process, produce the required vector operator.

Computational steps:

(1) The four collinearity equations are

$$\begin{aligned} k_x\mathbf{k}\cdot\mathbf{P}_m + \mathbf{i}\cdot\mathbf{P}_m &= 0 \\ k_y\mathbf{k}\cdot\mathbf{P}_m + \mathbf{j}\cdot\mathbf{P}_m &= 0 \\ k'_x\mathbf{C}_3\cdot\mathbf{P}_m + \mathbf{C}_1\cdot\mathbf{P}_m - k'_x\mathbf{C}_3\cdot\mathbf{b} - \mathbf{C}_1\cdot\mathbf{b} &= 0 \\ k'_y\mathbf{C}_3\cdot\mathbf{P}_m + \mathbf{C}_2\cdot\mathbf{P}_m - k'_y\mathbf{C}_3\cdot\mathbf{b} - \mathbf{C}_2\cdot\mathbf{b} &= 0 \end{aligned} \quad (2)$$

To evaluate Equation 2, we construct the apparent direction vectors ($\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{C}_3$) from the relative orientation parameters and scale each image vector data by the inverse of the camera constant: i.e.,

$$k_x = \frac{x_a}{c},\ k_y = \frac{y_a}{c}$$

$$k'_x = \frac{x'_a}{c},\ k'_y = \frac{y'_a}{c}$$

(2) We compute the design matrices with respect to the vector $\mathbf{P}_m$ by applying vector differential operators to Equations 2 and noting that

$$\frac{\partial P_m}{\partial x_m} = i,\ \frac{\partial P_m}{\partial y_m} = j,\ \frac{\partial P_m}{\partial z_m} = k$$

and

$$\mathbf{i}\cdot\mathbf{j} = \mathbf{i}\cdot\mathbf{k} = \mathbf{j}\cdot\mathbf{k} = 0,$$

we have

$$\mathbf{A}_1 = \frac{c}{\mathbf{k}\cdot\mathbf{P}_m}\begin{bmatrix} \mathbf{i}\cdot\mathbf{i} & 0 & k_x\mathbf{k}\cdot\mathbf{k} \\ 0 & \mathbf{j}\cdot\mathbf{j} & k_y\mathbf{k}\cdot\mathbf{k} \end{bmatrix}$$

$$f_1 = -\frac{c}{\mathbf{k}\cdot\mathbf{P}_m}\begin{bmatrix} k_x\mathbf{k}\cdot\mathbf{P}_m + \mathbf{i}\cdot\mathbf{P}_m \\ k_y\mathbf{k}\cdot\mathbf{P}_m + \mathbf{j}\cdot\mathbf{P}_m \end{bmatrix}$$

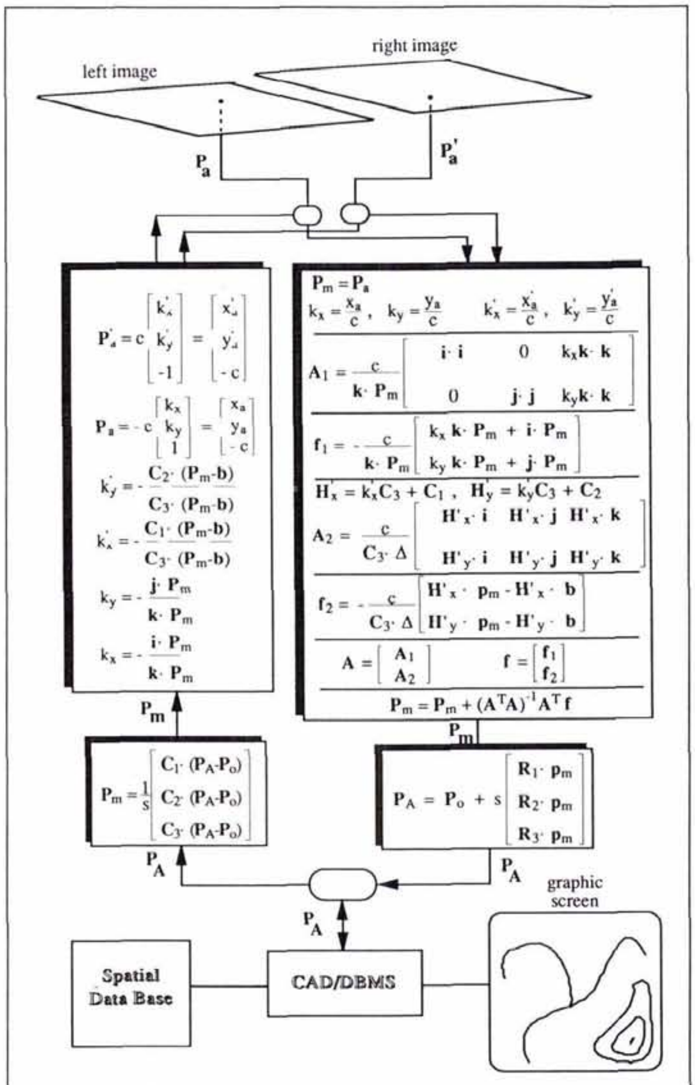
right image
left image
P_a
P'_a
P_m
P_A
graphic screen
Spatial Data Base
CAD/DBMS
$\mathbf{P}_m = \mathbf{P}_m + (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \mathbf{f}$

Figure 3. System design based on image space intersections.

$$\mathbf{H}'_x = k'_x\mathbf{C}_3 + \mathbf{C}_1, \qquad \mathbf{H}'_y = k'_y\mathbf{C}_3 + \mathbf{C}_2$$

$$\mathbf{A}_2 = \frac{c}{\mathbf{C}_3\cdot\Delta}\begin{bmatrix} \mathbf{H}'_x\cdot\mathbf{i} & \mathbf{H}'_x\cdot\mathbf{j} & \mathbf{H}'_x\cdot\mathbf{k} \\ \mathbf{H}'_y\mathbf{i} & \mathbf{H}'_y\cdot\mathbf{j} & \mathbf{H}'_y\cdot\mathbf{k} \end{bmatrix}$$

$$f_2 = -\frac{c}{\mathbf{C}_3\cdot\Delta}\begin{bmatrix} \mathbf{H}'_x\cdot\mathbf{P}_m - \mathbf{H}'_x\cdot\mathbf{b} \\ \mathbf{H}'_y\cdot\mathbf{P}_m - \mathbf{H}'_y\cdot\mathbf{b} \end{bmatrix}$$

$$\Delta = \mathbf{P}_m - \mathbf{b}$$

so that for the two rays we have

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}, f = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

and from an application of least squares, we obtain the iterative operator

$$\mathbf{P}_m^{(i)} = \mathbf{P}_m^{(i-1)} + (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T f. \qquad (3)$$

The approximate model space vector required to start the iteration may be taken as the left image vector for the point. Hence, for system implementation with this approach, Equation 3 is the iterative M-processor algorithm required. This has been used in the prototype design shown in Figure 3.

Algorithm Based on Bisection of Parallax Vector

This performs a **C**-space to **R**-space rotation of two corresponding image space vectors into the model space; it then computes the shortest distance between the rotated corresponding rays and derives the model position vector by adding half of the distance to the first ray. The method uses direction vectors of the two rays to compute the minimum parallax vector using a quasi least-squares approach after Cooper (1987). Basically, the method projects the left ray onto the natural or real axes of the model space (a consequence of the dependent pair relative orientation with model axes coinciding with the left image system), and projects the right ray onto the apparent direction vectors of the model space. The parallax vector bisection algorithm given below is for the dependent pair relative orientation.

Computational steps:

(1) Construct the apparent direction vectors of the **R**-space, i.e., $\mathbf{R}_1$, $\mathbf{R}_2$, $\mathbf{R}_3$ using the parameters of relative orientation and compute the following (note that $\mathbf{P}_a$ and $\mathbf{P}'_a$ are corresponding image space vectors and $\mathbf{R}_1$, $\mathbf{R}_2$, $\mathbf{R}_3$ are the same as the row of vectors of a 3D rotation matrix in analytical geometry):

$$\mathbf{P}_a = \begin{bmatrix} x_a \\ y_a \\ -c \end{bmatrix}, \mathbf{P}'_a = \begin{bmatrix} x'_a \\ y'_a \\ -c \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$$

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{i}\cdot\mathbf{P}_a \\ \mathbf{j}\cdot\mathbf{P}_a \\ \mathbf{k}\cdot\mathbf{P}_a \end{bmatrix}, \mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1\cdot\mathbf{P}'_a \\ \mathbf{R}_2\cdot\mathbf{P}'_a \\ \mathbf{R}_3\cdot\mathbf{P}'_a \end{bmatrix}$$

$$\overline{\mathbf{T}}_1 = \frac{\mathbf{T}_1}{|\mathbf{T}_1|}$$

$$\overline{\mathbf{T}}_2 = \frac{\mathbf{T}_2}{|\mathbf{T}_2|}$$

$$D = 1 - (\overline{\mathbf{T}}_1\cdot\overline{\mathbf{T}}_2)^2$$

$$s_1 = \frac{\overline{\mathbf{T}}_1\cdot\mathbf{b} - (\overline{\mathbf{T}}_1\cdot\overline{\mathbf{T}}_2)\overline{\mathbf{T}}_2\cdot\mathbf{b}}{D}$$

$$s_2 = \frac{(\overline{\mathbf{T}}_1\cdot\overline{\mathbf{T}}_2)\overline{\mathbf{T}}_1\cdot\mathbf{b} - \overline{\mathbf{T}}_2\cdot\mathbf{b}}{D}$$

(2) The model coordinate vector is then computed as

$$\mathbf{P}_m = \frac{\mathbf{b} + s_1\overline{\mathbf{T}}_1 + s_2\overline{\mathbf{T}}_2}{2} \qquad (4)$$

This algorithm is used in the system design shown in Figure 4.

Algorithm Based on y-Parallax Averaging

This method is similar to that for the parallax vector bisection except that the parallax vector is assumed to have only a *y*-component, i.e., the lack of intersection occurs only along the *y*-direction in the model space. The last few lines of the computational steps in the previous section are changed to

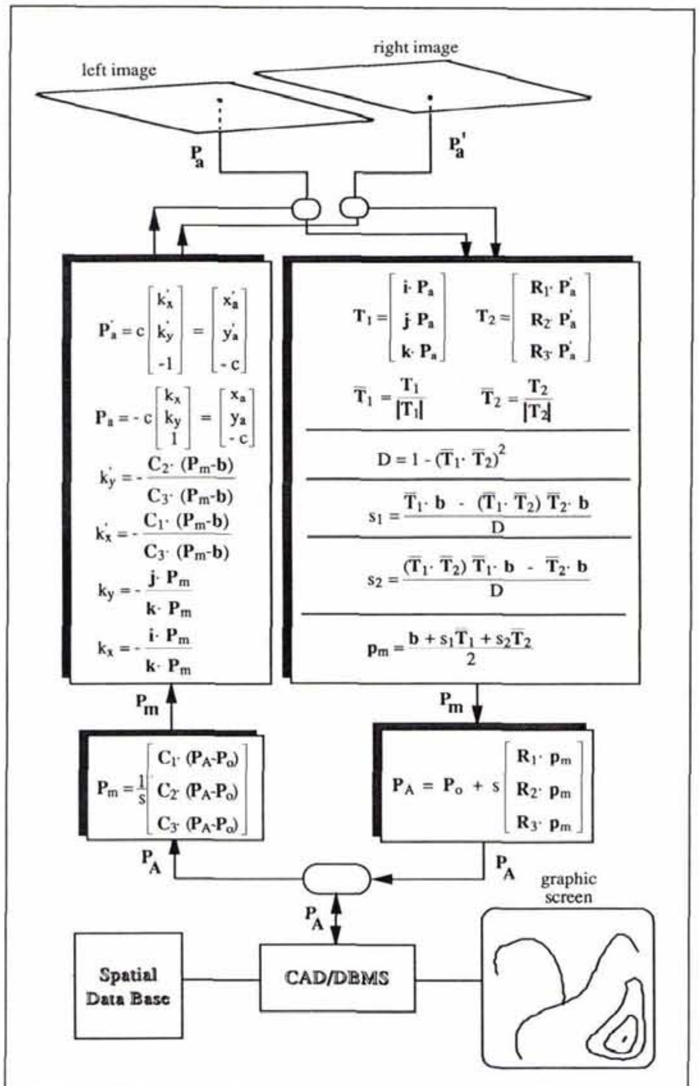

Figure 4. System design based on model space parallax vector bisection.

$$D = 1 - (\overline{\mathbf{T}}_1 \cdot \overline{\mathbf{T}}_2)^2$$

$$s_1 = \frac{\overline{\mathbf{T}}_1 \cdot \mathbf{b} - (\overline{\mathbf{T}}_1 \cdot \overline{\mathbf{T}}_2)\overline{\mathbf{T}}_2 \cdot \mathbf{b}}{D}$$

$$s_2 = \frac{(\overline{\mathbf{T}}_1 \cdot \overline{\mathbf{T}}_2)\overline{\mathbf{T}}_1 \cdot \mathbf{b} - \overline{\mathbf{T}}_2 \cdot \mathbf{b}}{D}$$

$$\mathbf{P} = \mathbf{b} - s_1\overline{\mathbf{T}}_1 + s_2\overline{\mathbf{T}}_2$$

$$\mathbf{P}_y = \begin{bmatrix} 0 \\ \mathbf{P} \cdot \mathbf{j} \\ 0 \end{bmatrix}$$

Hence, the M-processor may be written as

$$\mathbf{P}_m = s_1\overline{\mathbf{T}}_1 + \frac{1}{2}\mathbf{P}_y \tag{5}$$

This has been used in the design shown in Figure 5.

Algorithm for the Inverse M-Processor

Conceptionally, the task here is to resolve the model space vector element into its left and right image space vector elements. This involves a projection of the model space vector $\mathbf{P}_m$ (Figure 2b) onto the appropriate image space axes, which in the ARDOVS theory is an **R**-space to a **C**-space operation. One way to derive this algorithm is to simplify the **C**-space collinearity Equations 1 or 2 to obtain the constant multipliers k_x, k_y for both the left and the right image cases.

For the left image, we have

$$k_x = -\frac{\mathbf{i} \cdot \mathbf{P}_m}{\mathbf{k} \cdot \mathbf{P}_m}$$

$$k_y = -\frac{\mathbf{j} \cdot \mathbf{P}_m}{\mathbf{k} \cdot \mathbf{P}_m}$$

$$\mathbf{P}_a = -c \begin{bmatrix} k_x \\ k_y \\ 1 \end{bmatrix} = \begin{bmatrix} x_a \\ y_a \\ -c \end{bmatrix} \tag{6}$$

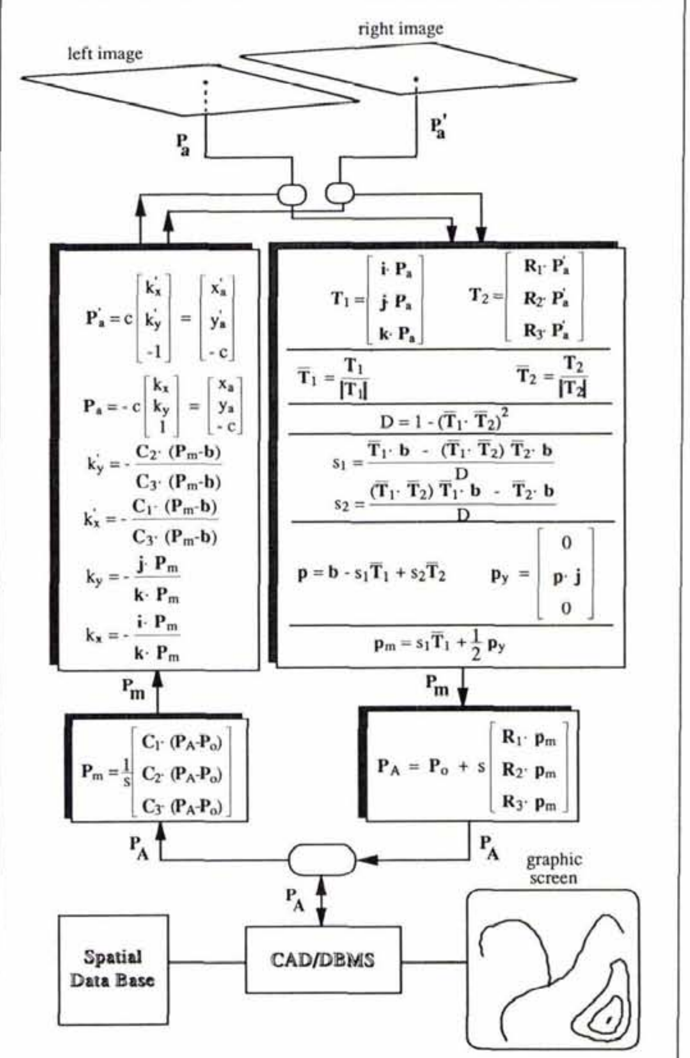

Figure 5. System design based on model space *y*-parallax averaging.

and, for the right image, we have

$$k'_x = -\frac{\mathbf{C}_1\cdot(\mathbf{P}_m - \mathbf{b})}{\mathbf{C}_3\cdot(\mathbf{P}_m - \mathbf{b})}$$

$$k'_y = -\frac{\mathbf{C}_2\cdot(\mathbf{P}_m - \mathbf{b})}{\mathbf{C}_3\cdot(\mathbf{P}_m - \mathbf{b})}$$

$$\mathbf{P}'_a = c\begin{bmatrix} k'_x \\ k'_y \\ -1 \end{bmatrix} = \begin{bmatrix} x'_a \\ y'_a \\ -c \end{bmatrix} \qquad (7)$$

This inverse M-processor algorithm has been used in the system designs of Figures 3, 4, and 5.

Experimental Investigation

On the one hand, this experiment was designed to determine the amount of distortion introduced into the data conversion process by each of the M-processors in the course of a round-loop data processing (i.e., image to model, model to object, object to model, and model to image data conversions). On the other hand, it was to ascertain their relative speed of operation in the process. The underlying idea of the test was that, because the three system configurations given above differ only in the M-processor, any differences in their performance are attributable to the different algorithms used for the M-processor.

Three data sets were employed for the test. The first dataset was a simulated block of four stereopairs generated with micrometre accuracy. The second dataset was a subset of four stereopairs from the Edmundston Block which covers an area in north western New Brunswick. The selection was based on a previous study that showed points in these models to be of higher accuracy than the average for the whole block (Moniwa, 1977). The third dataset was the entire Edmundston Block of 12 stereopairs. The relative and absolute orientation elements for each dataset were computed by an off-line module prior to the treatment of the data, and these values were held fixed in the three systems. Each set was passed through each system and processed in batch mode, one model at a time, from the image space through the model space to the object space (the object space is represented by the database in Figures 3, 4, and 5), and through the reverse operation the data were returned to the image space. The returned data were then compared with the original and the differences recorded. Furthermore, the times for processing each data set were recorded, and from these, the average time (cycle time) taken to convert one point by each processor was determined.

Results

Table 1 shows the RMSE of the differences between the input and the recomputed image points (in µm) averaged over all the points in each block for both the left and the right images. Table 2 shows the average time (in milliseconds) for converting one point from the image space to the model space and back to the image space for the three processors. Figure 6 is a graphical illustration of the data in Table 1. It shows that the distortions from the three processors are practically the same, and that the distortions increase with the level of inaccuracies of measurement. Figure 7 shows graphically the average time taken for both forward and the reverse conversion for one point. It is seen that the time for the image space intersection is about three times more than for the other two methods, and that the *y*-parallax method used the least time. The data in Table 3 show the variation of the distortions introduced into the left image and the right image by each of the three processors. As expected, the first and second processors introduce equal distortions on both the left and the right images, whereas the third is seen to introduce more to the right image than to the left image (see also Figure 8). However, this differential distortion will only be significant when the measurement noise is less than about 2 to

TABLE 1. RMSE OF DISTORTIONS IN IMAGE SPACE (µM)

Data set	sys1	sys2	sys3
1	0.1826	0.1846	0.1857
2	7.1257	7.1794	7.2712
3	12.7279	12.7988	12.8412

Note: sys1 refers to the algorithm based on image space intersection
sys2 refers to the algorithm based on bisection of the parallax vector
sys3 refers to the algorithm based on *y*-parallax averaging.

TABLE 2. FORWARD AND REVERSE CONVERSION TIME FOR ONE POINT (MS)

		sys1	sys2	sys3
Data set	1(80pts)	1390	450	440
	(1pt)	17.38	5.62	5.50
	2(69pts)	1200	380	380
	(1pt)	17.39	5.51	5.51
	3(151pts)	2640	870	850
	(1pt)	17.48	5.76	5.63
	avg./1pt(ms)	17.42	5.63	5.55

TABLE 3. RMSE OF LEFT AND RIGHT IMAGE SPACE DISTORTIONS (µM)

Data set	system	Left	Right
1	1	0.1814	0.1837
	2	0.1861	0.1830
	3	0.1588	0.2126
2	1	7.1225	7.1289
	2	7.1707	7.1882
	3	6.3252	8.2173
3	1	12.7162	12.7396
	2	12.8398	12.7577
	3	11.7179	13.9634

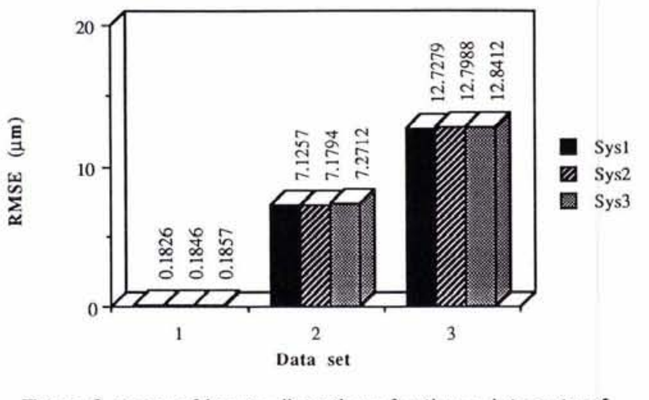

Figure 6. RMSE of image distortions for three data sets of different precision.

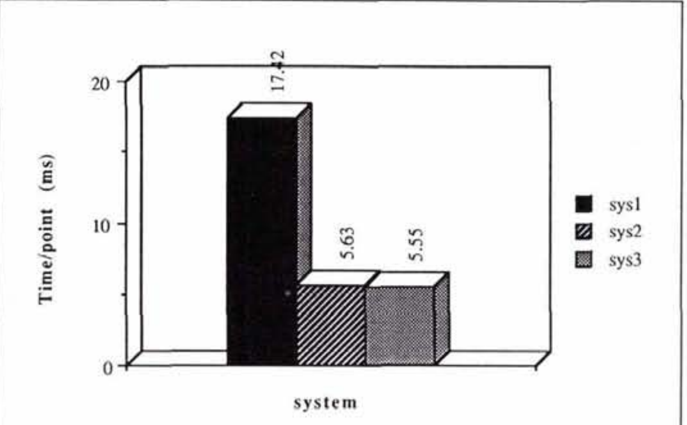

Figure 7. Time (ms) for forward and reverse conversion of one point.

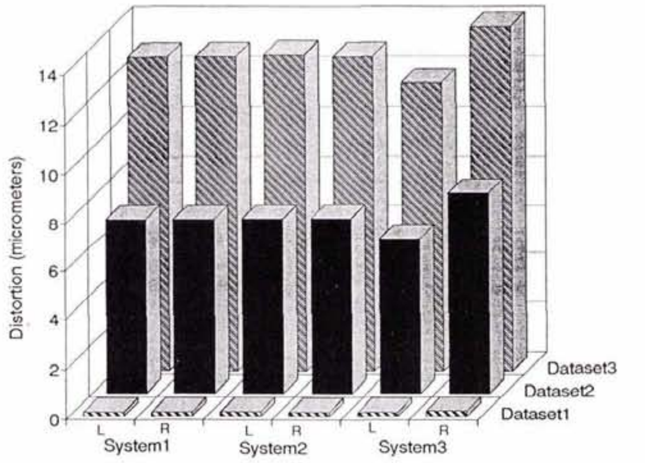

Figure 8. Variation of distortion of left and right photos for the three systems.

3 μm, and thus can be neglected for all practical purposes, which renders this less rigorous process still fully acceptable.

Closing Remarks

It follows from the results of this investigation that the three processors considered provide the same accuracy within practical limitations. This confirms results of other investigations in this area. Furthermore, the method of image space intersection provides the most rigorous solution but consumes the largest time; therefore, this method is not recommended for systems in which the requirement for speed is critical. On aggregate, it is concluded that an M-processor based on the parallax vector bisection algorithm will lead to the most efficient system implementation in a microcomputer-based environment.

References

Ackermann, F., 1984. Digital Image Correlation: Performance and Potential Applications in Photogrammetry, *Photogrammetric Record*, 11(64):429-439.

Cooper, M. A. R., 1987. *Control Surveys in Civil Engineering*, Nichols Publishing Company, New York.

Faig, W., 1984. *Aerial Triangulation and Digital Mapping*, Monograph 10, School of Surveying, University of New South Wales, Kensignton, NSW, Australia.

Ghosh, S. K., 1988. *Analytical Photogrammetry*, 2nd Edition, Pergamon Press, New York.

Hellwich, O., and W. Faig, 1992. Graph-Based Matching of Stereo Features, *International Archives of Photogrammetry and Remote Sensing*, 29(part B3):307-317.

Miles, M. J., 1968. The Theory of the Analytical Solution of the Stereogram, *Photogrammetric Record*, 6(32):196-201.

Miller, S. B., U. V. Helava, and K. Devenecia, 1992. Soft-Copy Photogrammetric Workstations", *Photogrammetric Engineering & Remote Sensing*, 58(1):77-84.

Moniwa, H., 1977. *Analytical Photogrammetric System with Self Calibration and its Applications*, Ph.D. Dissertation, Department of Surveying Engineering, University of New Brunswick, Fredericton, NB, Canada.

Olaleye, J., 1992. *An Investigation into the Optimum Software Architecture for an Analytical Photogrammetric Workstation and its Integration into a Spatial Information Environment*, Ph.D. Dissertation, Department of Surveying Engineering, University of New Brunswick, Fredericton, NB, Canada.

Schenk, T., and C. K. Toth, 1992. Conceptual Issues of Softcopy Photogrammetric Workstations, *Photogrammetric Engineering & Remote Sensing*, 58(1):103-112.

Schut, G. H., 1973. *An Introduction to Analytical Strip Triangulation, with a Fortran Program*, Revised Edition, Division of Physics Publication, NRC-13148, National Research Council of Canada.

(Received 17 June 1992; accepted 23 September 1993; revised 7 December 1992)