HIERARCHICAL DATA STRUCTURES AND ALGORITHMS FOR DIGITAL STEREOSCOPICAL MENSURATION

Rune Larsson
Department of Photogrammetry
Royal Institute of Technology
Sweden
Commission III

ABSTRACT

A number of generalisations of existing hierarchical data structures are presented along with suggestions for algorithms for access to image data in varying resolution levels. The data structures are used to form a base for a system for stereoscopical mensuration including schemes for data compression, filtering, enhancement, matching, feature detection, segmentation and scene analysis. A connection to three-dimensional object representation such as a Digital Elevation Model is discussed.

INTRODUCTION

In order to process digital images in computers, efficient algorithms and data structures must be utilized. In the design of such a system, consideration has to be taken to

- the format of input data from e.g. scanners and digital cameras
- the data structure most suitable for any operation on the digital images
- the possible output formats such as digital elevation models (DEM's), orthophotos and other.

During the processing, image data will be converted between different storage structures. Since the amount of data is large, it must be a goal to minimize the number of conversions and make them as efficient as possible.

In this paper a class of data structures will be presented. They will be evaluated with respect to their efficiency and usefulness for photogrammetric applications.

The design of data structures is closely connected with the algorithms that work upon the data. For that reason, a number of different operations will be discussed to show their influence.

HIERARCHICAL PICTURE INDICES AND DATA STRUCTURES

From the start it is possible to focus on hierarchical data structures and picture indices. They are designed for the access to images of varying resolution. It is easy to change between an overview, a region or a detail of an image. This is a basic property for many operations on images. A continous switching between detail study and overview is characteristic for the human vision. A hierarchical data structure allows for a generalisation of the image while simultaneously keeping the detailed original image. A lower resolution image is obtained by computing an averaged or median-filtered version of the original image. This can be repeated recursively nearly ad infinitum or until the image information is exhausted.

TO USE A PICTURE INDEX

A picture index is a data structure used for access to relevant image data. It is often separated from the stored image and can be compared to a tailored data base for the search of image patches within one or many scenes.

A large digital image, e.g. a digitized aerial photograph, may contain several million pixels. In order to process such images, they have to be divided into parts, e.g. line by line (like a TV picture) or in a number of image patches. The division is necessary for the processing of the image.

The picture index is the organisation of all parts of the image. It is designed to allow for rapid access to neighbour patches and to patches with higher or lower resolution. The search is made horisontally (for a neighbour) or vertically (for an other resolution).

Often large parts of the image is of no use to the current progess, e.g. those parts with no stereo cover. In that case it is unnecessary to store those parts and the picture index should be able to account for those parts as well.

To search a picture vertically is an ideal way of working. The possibility to "zoom in" at details is an efficient way of searching, both for the human and automatic system. A picture index will make such searches possible. Without that, one would be forced to start the search at the bottom, in the detailed, voluminous original picture.

The picture index can also be used at the planning stage of image processing. In stereo photogrammetry, much experience have been gained about how a stereo pair is evaluated. A hierarchical picture index is a powerful tool for the equivalent operations on digital pictures.

In interactive work, a hierarchical picture index is also valuable. The resolution of pictures presented on display units can be varied and in each moment be suited to the process. Picture generation at full resolution can be avoided, thus saving precious time. Also in transmission of picture data, time can be saved by sending low resolution overviews, which can be used to decide which image parts is needed in more detail. The latter is very important, since picture data easily can overload even high capacity communication channels.

DIFFERENT HIERARCHICAL PICTURE INDICES

Quadtrees

One type of picture index that conform to the needs shown above is the quadtree. It is a tree structure where each node has four children nodes. In a quadtree only the outermost nodes (the leaves) carry any information. It can be utilized in several ways. One way is to let each leaf represent a pixel. Four leaves can then represent a patch of 2 by 2 pixels, which also can be represented by their parent node. Nodes higher up in the hierarchy can represent larger patches. The root of the quadtree would then represent the entire image and their children each represent one image quadrant.

In a quadtree one must also know the position of the children. From a node, the children can be denoted NW, NE, SW and SE, i.e. the direction from parent to child.

The quadtree is suitable for binary images, where each pixel is either black or white. Many nodes can be eliminated from such a tree through eliminating all children groups in which all have the same colour. In that case the parent node can be assigned the colour of its children. The same reduction can be carried higher up in the hierarchy until all parents have children of both colours.

In figure 1 this picture index is shown for a patch with a binary edge. There are 41 out of the original 85 nodes in the tree after the reduction.

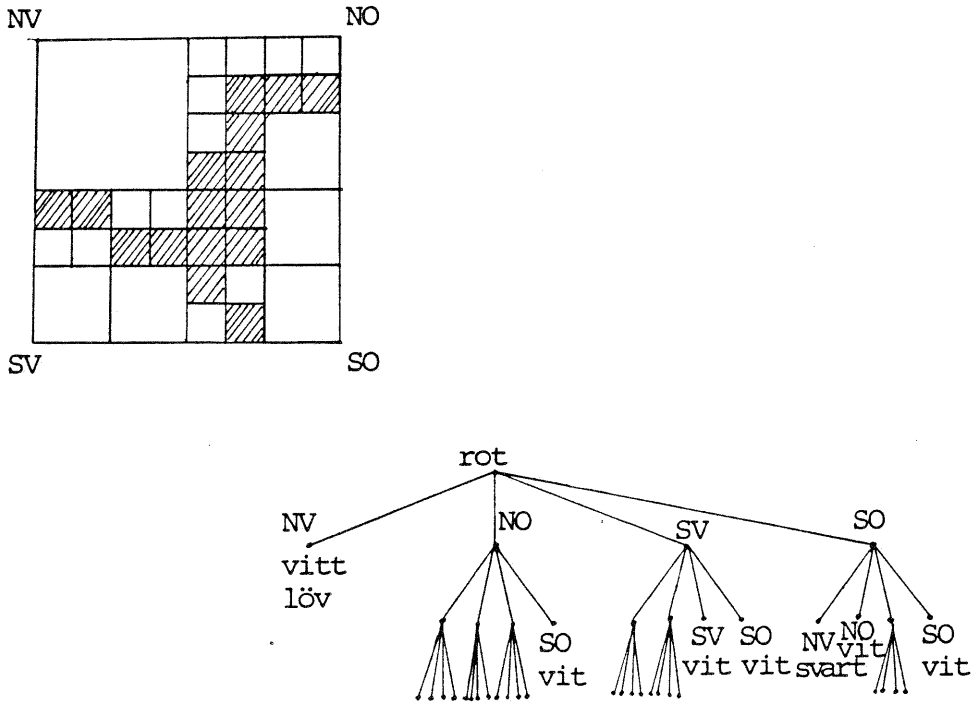

Figure 1 Quadtree

Such a reduction is not feasible with halftone pictures. The information is too complex to be reduced in such a simple way. However, many methods for matching are performed using edges derived from the original image. Picture indices for binary images is thus of interest also for our applications.

The leaves of a quadtree can also represent an image patch of a certain size. The patches can e.g. be stored in a matrix, the address of which is stored in the leaf. The neighbourhood relations are still retained, and it is the operations acting on the picture data that will be decisive in the choice of organisation.

Pyramids

The pyramid is another data structure suitable for the access of picture data (Tamimoto 1975). It shares some of the features of the quadtree but it is implemented differently. The original picture is the base of the pyramid,

stored e.g. in matrix form. From that, a series of picture matrices is created with successively decreasing resolution. From each 2 by 2 pixel patch, one pixel is created in the next lower reoslution. Each new picture matrix occupies thus only 25% of the storage of its predecessor. A complete pyramid is stored in 4/3 of the storage of the original picture.See figure 2.

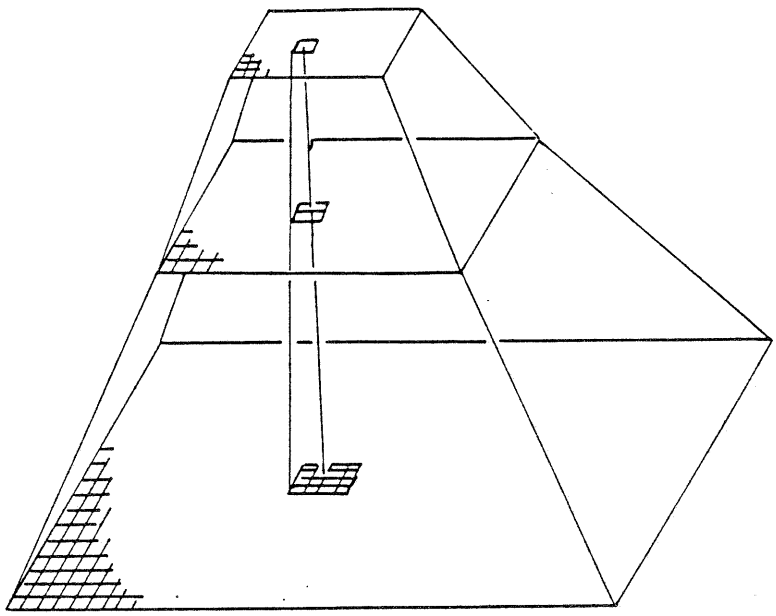

Figure 2 Pyramid

A subpicture at one level of a pyramid can be compared to the set of parents on a certain level in a quadtree. The two storage structures have much in common. A closer study of possible implementations can reveal more similarities.

Other Picture Indices

Under this heading, higher level picture indices are collected. With a higher level is understood, that the indices are used for picture structures or for semantic picture information. These indices are used in the analysis of picture contents, indicating that one or more levels of interpretation of content have been performed. In this paper though, the topic is aimed at interpretations at the low level, and thus these indices are more of interest in further research.

INPUT AND OUTPUT DATA STRUCTURE

The data for input and output of digital images are often structured line by line. This is true for most scanning equipment, whether they are scanning the scene directly or from a photograph. The lines cover most often the entire width of the scene and follow each other sequentially.

To complicate things, some equipment divide the image in strips or patches, which in turn are digitized line by line. Problems will then occur at the borders between the strips or patches, where mechanical imperfections will impair the stability of the relative positions of the strips. The border areas are often overlapped, i.e. digitized in both adjoining strips, but

often little or no correction is applied to bring the strips in one and the same co-ordinate system. Some kind of preprocessing must be made in order to obtain geometrically corrected digital images.

To convert digital image data to the internal data structure, which often is structured with patches, can be timeconsuming unless large buffer areas are available. A number of lines, possibly long, must be buffered before the patches can be assembled and stored away.

INTERNAL DATA STRUCTURE

The implementation of a suitable data structure for digital photogrammetry will be based on the earlier presented hierarchical data structures.

In photogrammetric applications, large format fotos are digitized and stored, and the amount of information in a stereo pair is such that special attention has to be given to aspects of efficiency and storage utilization.

For that reason, a pyramid organization seems favourable, in that single pixels can be accessed using ordinary addressing methods of matrices. The entire image can, however, seldom be stored in that way, since the addressing range of most computers is insufficient for that purpose.

One way of solving this is to decompose the image in "patches" of suitable and fixed size, and access the patches through a quadtree, where each leaf node contain a patch of the original image. The higher levels of the pyramid can be decomposed in patches of equal size which can be stored in the non-leaf nodes of the quadtree. Thus four adjacent leafpatches can be used to generate one patch of a lower resolution.

This operation of generating lower-resolution patches can be applied recursively up trough the levels of the quadtree. The set of nodes at a certain level of the quadtree corresponds to one level in the pyramid.

How can these patches be accessed from an application program? The simplest way is to allow access to a fixed number of connected patches, e.g. one parent patch and its four sons. They should be presented to the program (at least at a low level of abstraction) in five image matrices, one for the parent, and four for the children. See figure 3.

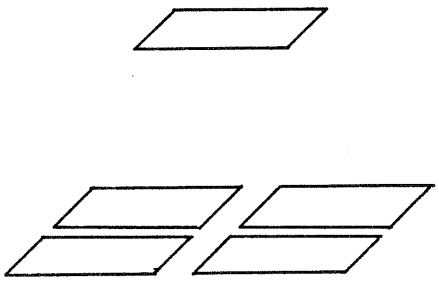

Figure 3 Pyramid window.

By appropriate subroutine calls this five pane window can be moved or stepped through the entire quadtree on demand from the program, both in lateral and vertical direction (Up, North, South, East, Down SW, Down SE, Down NW, Down NE).

Other designs of the window could of course be chosen, e.g. one central patch, eight neighbours, four children and a parent. The central idea is to keep the window fixed in design and move it as the process demands. The reason for simultaneous access to more than one reduction is that many operations on digital images utilize that feature.

Dynamic Creation of the Image Hierarchy

In photogrammetric evaluation of digital stereo images the processing must be separated in several stages. The operations can to a large extent be described in terms familiar to the photogrammetrist, namely interior, relative and absolute orientation. The heavy task is the derivation of three-dimensional co-ordinates, i.e. the matching of corresponding image details. The matching process is not described here, but since it is a computationally heavy task, measures must be taken to perform the matching as efficiently as possible. One way is to resample the images to normal case photography and to utilize matching along the epipolar lines.

This is done when the interior and relative orientation is computed. The resampling will lead to that the hierarchical data structure containing the original imagery is no longer of use. The detailed bottom image is totally resampled and higher level pictures of lower resolution must be computed anew.

In this case, the creation of the temporary first version of the image data should only be done on a "if needed" -basis. In other words, no parts of the pyramid should be computed unless they are needed.

What parts of the image are then needed to obtain the resampling data? The operations that are carried out for that determination are the interior and relative orientation. If the fiducials are of good contrast, the interior orientation can be performed using only the high resolution image (Mikhail 1983). The relative orientation, on the other hand, necessitates the matching of say, 15 relative orientation points. Since only crude approximate values for the matching is known, e.g. based on standard aerial photograph geometry, the matching points will be found quicker if lower resolution can be utilized.

There are two possible ways to avoid generating the entire hierarchy from the original picture content. One way is to manually indicate the relative orientation points in both pictures using an image display unit and a cursor, controlled by an operator. These measurements need then only be finely adjusted by the matching algorithm, e.g. by the method presented by Förstner (1982). That method can compute not only the translations, but also affine transformation parameters for the neighbourhood of each matched object. The affine parameter computation is, however, expensive in terms of number of arithmetic operations but since affine deformations can account for much of the local differences around the object, that extra work is well worth doing. The number of points to match in this way is also low, making the matches for the relative orientation feasible anyway.

The other way of avoiding the generation of the entire image hierarchy is to decrease the number of levels between the highest and the lowest

resolution. It is possible that only one or two lower-resolution levels are necessary for the automatic matching of the relative orientation points. It is possible to vary the resolution steps between the different levels in a pyramid and that may be used to obtain sufficiently good starting values for the final match. It remains to be determined how many and which resolution levels are needed to perform the relative orientation with a minimum of work.

How is then the entire image hierarchy created? It is possible to design the software interface to the pyramid in such a way that only the necessary patches of the pyramid are computed, i.e. those which are asked for. They are then, when created, stored on disc for later retrieval. In the case the lowest resolution patch is asked for, the entire pyramid must be computed, since that patch cover the image area. For the coding of the pyramid software, a language with the possibility of recursive calls must be used, since the tree-handling is of a recursive nature. For that reason, FORTRAN cannot be used.

Addressing Schemes

The hierarchical data structure is based on a modified quadtree, where each node contains one (quadratic) image patch of fixed size. Addressing within the patches are made using matrix addressing technique. The patch size is chosen such that the input/output operations can be performed effectively.

To minimize disc accesses the processing of the image can be studied and the software interface can be designed subsequently. There are many reasons to believe that the processing will typically access patches in the neighbourhood of the current patch or patches of the same area but in another resolution. In that case, a dynamic allocation facility for the recently used patches can be utilized. Should a patch be requested that is located in dynamic storage, a link to that patch is obtained without a disk access. When storage is exhausted, the space occupied by the least recently used patch is utilized for the next patch.

A suitable simple organisation for the patches can be made with the help of a heap. This data structure consists of a vector of patch addresses on disk. The first vector element contain the address to the root patch, i.e. the patch of lowest resolution which covers the entire image. The four elements following that contain the addresses of the children of the root, e.g. in the order (Nw, NE, SE, SW).

If the index in the heap vector for a patch is i, the index for the family is

```
ADR(SELF)   = HEAP(i)
ADR(PARENT) = HEAP((i+2)/4)
ADR(NW)     = HEAP(4*i-2)
ADR(NE)     = HEAP(4*i-1)
ADR(SE)     = HEAP(4*i)
ADR(SW)     = HEAP(4*i+1)
```

To find a neighbour, a more complicated address calculation is used. It consists of finding the child status (NW, NE, SE, SW) of the patch and if the wanted patch is outside the family, a recursive algorithm must be followed until the patch is found.

To avoid this, an index structure similar to the pyramid may be used. Let us assume that a pyramid (figure 2) is built where addresses to image patches are stored instead of individual pixel values. This pyramid may be traversed in different directions, both laterally and in depth. For each position in the pyramid, the corresponding image patch can be fetched from disk using the addresses in the pyramid.

In this way, lateral search for neighbours of the same resolution is easy to accomplish. It remains to be derived how vertical search, for parents and children, is performed. Figure 4 shows the index in binary of the individual elements of the three upper levels of a pyramid.

Level0(x,y)

x	0
y	0

Level1(x,y)

x	0	1
y	0	0
x	0	1
y	1	1

Level2(x,y)

x	00	01	10	11
y	00	00	00	00
x	00	01	10	11
y	01	01	01	01
x	00	01	10	11
y	10	10	10	10
x	00	01	10	11
y	11	11	11	11

Figure 4 Pyramid addressing.

To find the address of the father of a patch, the x- and y-indices (column and row index of the matrix) are simply shifted to the right one bit, which is equal to integer division by two. To find the NW son of a patch (!?), the indices are shifted left one bit, or multiplied by two. The other children are found by left shift of row and column index and incrementing

x index for the NE child,

y index for the SW child and

x and y index for the SE child.

To address the neighbours of an arbitrary patch, one is added to or subtracted from the x and y index as in ordinary matrix addressing.

Furthermore, the offset of the actual patch address from the beginning of each matrix is obtained by joining the x and y index values to one binary number. That number is the corresponding offset. This is because the dimension of each matrix in the pyramid is a power of two. Shifting e.g. the x index to the left the number of bits of the y index value is then equal to multiplication by the row (y) dimension of the matrix. Adding the shifted x index to the y index will then yield the final offset to the address of the patch on disk. If the image disk area is contigous, the offset could be used to compute the disk address directly. The suggested method is, however, more general and is to be preferred. It allows also for the storage of incomplete images, e.g. the photo areas with no stereo cover can be indicated by a zero or negative disk address. These areas need not be stored on disk at all.

EXAMPLE

Let two aerial photographs be digitized such that they are contained in a 4096 by 4096 pixel square. That corresponds to a pixel side of 60 μm in

the image. Only 60% of each photo has stereo cover and thus the bottom level of the two pyramids is stored in 19.5 Mbyte disk storage (assuming 8 bits per pixel). The storage for the two pyramids will then be 26 Mbyte (33% added for the lower resolution images). The number of patches and the number of elements of the two index pyramids, which is contained in RAM, will then depend on the patch size. For a patch size of 64 by 64 pixels, the number of patches will be 6656. Using a 32 bit disk address, the size of the pyramid index in RAM is 26 kbytes.

DISCUSSION

To search for something in a digital image is a timeconsuming task. By replacing ordinary search with a search in a pyramid, you replace the linear search with binary search. The search is started at a low resolution, and by successively increasing the resolution, the search is performed with equal precision but in a much more effective way.

The goal of this work is to design a tool for manipulating large digital images of varying resolution. An effective picture index is found, which is capable of handling the large amounts of data extracted from aerial photography.

Much more can be said about matching, and in general, digital image processing techniques applied to photogrammetry. The application closest at hand is the production of digital orthophotos using automated matching to assess terrain height. The resampling to an orthophoto is made using a DEM constructed from the matching. The pyramid is suitable for such an approach. The DEM can be stored in a data structure similar to the pyramid, as indeed all land information can be. The matching process is then started at low resolution, creating a coarse DEM, which is successively densified as the resolution level is increased. Starting values for "next" level can be obtained through interpolation in the coarse DEM, and the result is immediately used to improve the DEM.

The result of each match can also be checked for conformity with the DEM and tests can be applied to spot erroneous matchings. A failure to match properly can lead to that another method is chosen for the matching. The texture can also be analyzed and that result can be used to govern the method and parameters for the matching. There is a vast field of research which appears within digital image processing applied to photogrammetry and land information systems.

REFERENCES

Förstner W. 1982. On the Geometric Precision of Digital Correlation. International Archives of Photogrammetry, Vol 24-III, pp 176-189, Otaniemi, Finland 1982

Mikhail E M. 1983. Detection, and Sub-Pixel Location of Photogrammetric Targets in Digital Images. Presented at the Specialist Workshop on Pattern Recognition in Photogrammetry, September 1983, Graz. To be published in Photogrammetria

Tanimoto S, Paulidis T. 1975. A hierarchical data structure for image processing. Computer Graphics and Image Processing No 4, 1975.