Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm

GUNILLA BORGEFORS

Abstract—**Matching is a key problem in digital image analysis and edges are perhaps the most important low-level image features. Thus good edge matching algorithms are important. This paper presents such an algorithm, the hierarchical chamfer matching algorithm. The algorithm matches edges by minimizing a generalized distance between them. The matching is performed in a series of images depicting the same scene, but in different resolutions, i.e., in a resolution pyramid. Using this hierarchical structure reduces the computational load significantly. The algorithm is reasonably simple to implement, and it will be shown that it is quite insensitive to noise and other disturbances. This new matching algorithm has been tested in several applications. Two of them will be briefly presented here. In the first application the outlines of common tools are matched to gray-level images of the same tools. Overlapping occurs. In the second application lake edges from aerial photographs are matched to lake edges from a map. Translation, rotation, scale, and perspective changes occur. The hierarchical chamfer matching algorithm gives correct results using a reasonable amount of computational resources in all tested applications.**

Index Terms—**Aerial image registration, camera model, digital image processing, distance transformations, edge matching, resolution pyramids, robot vision.**

I. Introduction

Matching is a key problem in computer vision, image analysis, and pattern recognition: objects, units, or other features in the image must be recognized and named. Often the exact position of those entities must also be known. The problem is usually aggravated by noise and unavoidable errors in the preprocessing of the images, e.g., segmentation and edge extraction. Differences between images due to translation, rotation, scaling, and perspective must often be taken into account.

Matching methods can be loosely divided into three classes: algorithms that use the image pixel values directly, e.g., correlation methods; algorithms that use low-level features such as edges and corners; and algorithms that use high-level features such as identified (parts of) objects, or relations between features, e.g., graph-theoretic methods.

Methods that use the image pixel values directly are very sensitive to any change between images, e.g., a modest shift in illumination may make matching between otherwise equal scenes impossible. The same structures in images from different sensors cannot be identified. High-level matching methods are very insensitive to these disturbances. The drawback is that high-level features must first be extracted and identified and that is, in most cases, a difficult matching problem in itself.

The matching method presented here, called the hierarchical chamfer matching algorithm (HCMA), is on an intermediate level. It matches edge points or other low-level feature points, extracted from the digital images. The HCMA does not only determine the best match (or no match) and the corresponding position in the image, but it also gives a measure of the goodness of the match that is easy to interpret.

As the HCMA matches edge points, the images to be matched must first be preprocessed by an edge extraction algorithm. There are many such algorithms, but the ideal one, that finds all edge points but no non-edge points, is yet to be discovered. (Indeed that may not be possible. It may not even be possible to define the desired output of such an algorithm in a reasonable way.) Thus any edge matching algorithm must be able to handle imperfect data. Edge extraction will not be discussed further here. We suppose that an adequate algorithm is available.

Chamfer matching was first proposed in 1977 [2]. It is a technique for finding the best fit of edge points from two different images, by minimizing a generalized distance between them. The edge points of one image are transformed by a set of parametric transformation equations, that describe how the images can be geometrically distorted in relation to one another. The original version of the chamfer matching algorithm is useful only in a limited number of applications. It is a fine-matching method, that needs a good start hypothesis of the transfomation that brings the edges into correspondence. No further work on the chamfer matching seems to have been published (except [3]). However, the original idea has several good properties, the most important one being the ability to handle imperfect (noisy, distorted, etc.) data.

The original chamfer matching idea has now been developed into a universally useful edge matching algorithm. First the matching measure, i.e., the measure of correspondence between the pattern to be matched, has been improved [3]. The result of this improvement is

Manuscript received December 30, 1986; revised May 5, 1987.

The author is with the Swedish Defence Research Institute, Box 1165, S-581 11 Linköping, Sweden.

IEEE Log Number 8823603.

0162-8828/88/1100-0849$01.00 © 1988 IEEE

fewer false matches. The improved algorithm is presented in Section II. The second more important improvement is that the algorithm has been imbedded in a resolution pyramid. The matching is done not only in the original image resolution, but in a series of images, where each image is a representation of the original scene at a lower resolution. The matching will start in very low resolution and the low resolution results will guide the computations at finer levels. This multiresolution approach will speed up the computations considerably, so much so that matching problems that could not be solved with the single resolution algorithm (as the computational load was too heavy) can now be solved using quite moderate computation times. Large parts of the computations can be performed in parallel. The HCMA can thus be implemented in a very efficient way, using hardware capable of parallel computations. However, even the unsophisticated implementation used to develop and investigate the algorithm is not prohibitively slow. The hierarchical algorithm is described in Section III.

The algorithm can be adapted to different matching tasks. One type of application is image to image matching. Examples are the first step in a stereo algorithm, analysis of image sequences, and change detection.

Another type of application is template to image matching: identifying and locating prespecified objects in digital images. One example, recognition of common tools, is shown in Section IV. The camera is fixed. Thus, only translation and rotation of the templates are necessary. Overlapping occurs. Other examples of applications of template to image matching are target detection and identification, robot vision, and identification of organs or certain types of cells in biomedical imagery.

Yet another type of application is image to symbolic image matching (i.e., matching two very different images depicting the same scene). This type of application is illustrated in Section V, where the example is registration of aerial photographs. Registering an aerial image is equivalent to transforming it into the map coordinate system. The matching algorithm is used to identify the transformation. The photograph is distorted, not only by translation and rotation, but also by scale and perspective changes. All these six degrees of freedom, together with noise, complicate the matching. If the algorithm performs well in this complex case (which it does) it will also perform well in many other applications. Other examples of image to symbolic image matching are navigation using terrain data, and presentation and interpretation of multisensor data.

II. Basic Concepts

The basic algorithm, used at each resolution level, will be described in this section.

Two binary images, consisting of feature and nonfeature points, are to be matched. The feature can be any feature visible in both images, e.g., edges (used here), corners, bright spots, or areas with a certain texture. The two images are not treated symmetrically by the HCMA. For reasons that soon will become apparent one image is called the predistance image and the other the prepolygon image. In some applications, the assignation of the two images is arbitrary. In other applications a certain choice may be advantageous, or even necessary.

A. Distance Transformation

In the predistance image, each non-edge pixel is given a value that is a measure of the distance to the nearest edge pixel. The edge pixels get the value zero. The true Euclidean distance is resource demanding (time, memory) to compute, therefore an approximation is used. The operation converting a binary image to an approximate distance image is called a distance transformation (DT). It is important that the DT used in the matching algorithm is a reasonably good approximation of the Euclidean distance, otherwise the discriminating ability of the matching measure, computed from the distance values, becomes poor.

The DT used in the HCMA has been developed by the author; see [4], [6]. This DT uses iterated local operations. The basic idea is that global distances in the image are approximated by propagating local distances, i.e., distances between neighboring pixels, over the image. The propagation of local distances can be done either in parallel or sequentially. Sequential DT's are known as "chamfer" distances, hence "chamfer matching." A 3 × 3 pixel neighborhood is used. The two local distances in a 3 × 3 neighborhood are the distance between horizontal/vertical neighbors and between diagonal neighbors. In the original algorithm the values 2 and 3 were used, respectively. With these values the maximum difference from the Euclidean distance becomes about 13 percent. If the values 3 and 4 are used instead, the maximum difference is reduced to 8 percent. The well-known city block distance [11] has a maximum difference of 59 percent [6].

In [3] the city block and the 3-4 DT were compared to the Euclidean distance. The 3-4 DT was shown to be good enough, whereas the city block DT was not. Using the Euclidean distance itself is usually not necessary, as the edge points are influenced by noise. It is a waste of effort to compute exact distances from inexact edges.

In the binary edge image each edge pixel is first set to zero and each non-edge pixel is set to infinity. If the DT is computed by parallel propagation of local distances, then at each iteration each pixel obtains a new value using the expression:

$$\begin{aligned} v_{i,j}^{k} = \text{minimum}\,(& v_{i-1,j-1}^{k-1} + 4,\ v_{i-1,j}^{k-1} + 3, \\ & v_{i-1,j+1}^{k-1} + 4, \\ & v_{i,j-1}^{k-1} + 3,\ v_{i,j}^{k-1},\ v_{i,j+1}^{k-1} + 3, \\ & v_{i+1,j-1}^{k-1} + 4,\ v_{i+1,j}^{k-1} + 3, \\ & v_{i+1,j+1}^{k-1} + 4), \end{aligned} \tag{1}$$

where $v_{i,j}^k$ is the value of the pixel in position (i, j) at iteration k. The iterations continue until no value changes. The number of iterations is proportional to the longest distance occurring in the image.

The sequential DT algorithm also starts from the zero/infinity image. Two passes are made over the image, first "forward" from left to right and from top to bottom; and then "backward" from right to left and from bottom to top:

```
Forward:
for $i = 2, \cdots$, rows do
for $j = 2, \cdots$, columns do
    $v_{i,j}$ = minimum $(v_{i-1,j-1} + 4, v_{i-1,j} + 3,$
        $v_{i-1,j+1} + 4, v_{i,j-1} + 3, v_{i,j})$.
Backward:
for $i$ = rows $- 1, \cdots, 1$ do
for $j$ = columns $- 1, \cdots, 1$ do
    $v_{i,j}$ = minimum $(v_{i,j}, v_{i,j+1} + 3,$
        $v_{i+1,j-1} + 4, v_{i+1,j} + 3, v_{i+1,j+1} + 4)$.
                                                        (2)
```

B. Matching Measure

In the prepolygon image the edge pixels are extracted and converted to a list of coordinate pairs, each pair being the row and column numbers of an edge pixel. From this list the edge points that are actually used are later chosen according to some criterion, which is application dependent. The list of chosen points is henceforth called the *polygon*, even though the points may be scattered or representing several polygon segments.

The polygon is superimposed on the distance image; see Fig. 1. An average of the pixel values that the polygon hits is the measure of correspondence between the edges, called the *edge distance*. A perfect fit between the two edges will result in edge distance zero, as each polygon point will then hit an edge pixel. The actual matching consists of minimizing this edge distance. To make this minimization as simple as possible, the edge distance function should be as smooth and convex as possible. Also, the optimal position must, in some sense, really be the position of best fit..

The root mean square average (abbreviated r.m.s.) was chosen for the matching measure:

$$\frac{1}{3}\sqrt{\frac{1}{n}\sum_{i=1}^{n} v_i^2}, \tag{3}$$

where v_i are the distance values and n the number of points in the polygon. The average is divided by three to compensate the unit distance three in the DT. Note that the value actually used in the algorithm is the sum of the squares, which is integer. The real number is computed only for the convenience of the interpreter of the results and to compare matches where the number of polygon points are different.

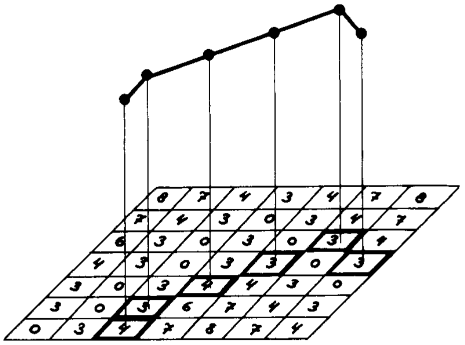

Fig. 1. Computation of the edge distance. The polygon representing an edge is placed over the distance image. The r.m.s. average of the pixel values that are hit, divided by 3, is the edge distance, in this case 1.12.

The original chamfer matching algorithm used the arithmetic average. Our choice of the r.m.s. is explained in [3]. There four different "averages" were compared: median, arithmetic, r.m.s., and maximum. The r.m.s. was found to give significantly fewer false minima in the edge distance function than any of the other averages. The test was admittedly heuristic, but quite extensive and hopefully general enough. To give a theoretical proof of the superiority of the 3-4 DT and the r.m.s. is probably impossible, as the number of matching configurations is infinite.

C. Optimal Positions

Each position of the polygon, determined by the transformation equations, corresponds to an edge distance. The position with the minimal edge distance is defined as the position with the best fit. The transformation equations that change the position of the polygon points must be parametric, i.e., the polygon position is described by a number of parameter values. In the simplest case the transformation is translation. A common case is translation and rotation, i.e., three parameters. Let (x, y) be the polygon coordinates and (X, Y) the position in the distance image. The transformation equations for translation and rotation become:

$$X = c_X + \cos(\text{rot})x - \sin(\text{rot})y$$

and

$$Y = c_Y + \sin(\text{rot})x + \cos(\text{rot})y, \tag{4}$$

where rot is the rotation angle and c_X and c_Y are the translation parameters in the X- and Y-directions, respectively. The (X, Y) coordinates are usually not integers. Thus, they must be rounded to the nearest integer values.

Finding the optimal polygon position is equal to finding the global minimum of a multidimensional function, where each dimension corresponds to a parameter in the polygon transformation equations. Unless the matching situation is very simple, the multidimensional function

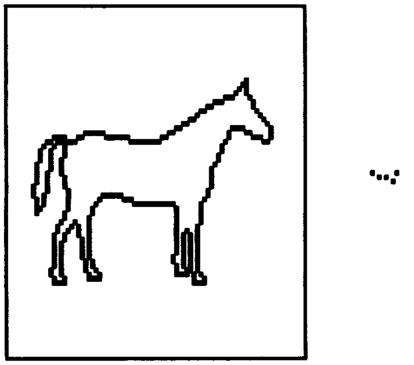

Fig. 2. A toy example to illustrate the matching algorithm. The five-point configuration to the right is searched for in the horse image (85 × 85 pixels).

that is to be minimized will have many local minima aside from the global one. A toy example has been prepared to illustrate this difficulty. Consider the outline of a horse to the left in Fig. 2. A polygon consisting of five points, to the right in Fig. 2, was matched to this image. The polygon represents the horse's nose. A distance image was computed from the horse's outline. The edge distance was computed for every possible translational polygon position within the distance image. In Fig. 3 all local minima of the two-dimensional edge distance function have been identified (black dots). There is only one position with perfect fit (the nose), but there are local minima all along the horse's edge. All these local minima will disturb any search algorithm used to find the global minimum. The total number of minima in Fig. 3 is 65. If the original matching measure, chamfer 2-3 DT and arithmetic average, would have been used, then the number of local minima would have been 115.

It must be pointed out that this toy example is more difficult than almost any realistic example. Five points from a very small area carry too little information. If more points were searched for, or the points more spread out, the number of local minima would decrease dramatically. However, the example does illustrate the difficulties that appear even in realistic applications. The properties of the edge distance function in general are such that to find the global minimum, the minimization must be started very close to the optimal position. Thus, searches must be started in a large number of different positions, so that at least one of them is close enough to the optimal position.

The averaging in the computation of the edge distance makes the matching measure relatively tolerant to minor differences between edges. However, when the matching is inexact, the global minimum must occur at a position that can be accepted as actually being the best match. Fig. 4 shows four examples of resulting optimal matches. The continuous lines are lake edges from a map and the dotted lines are noisy lake edges extracted from an aerial photograph (See Section V for details of this application). The figure shows that the optimal positions, found with reasonable effort, are acceptable matches, even though the dotted edges are severely disturbed (parts missing and broken up into several contours).

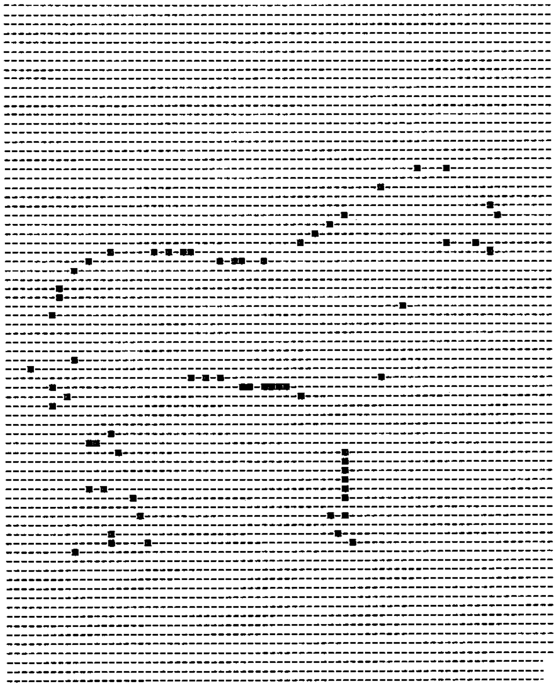

Fig. 3. The local minima of the edge distance function for the example in Fig. 2. The global minimum is the one represented by a bigger patch.

III. Hierarchical Algorithm

This section describes the imbedding of the chamfer matching algorithm into a hierarchical structure, a resolution pyramid. A resolution pyramid includes not only the original digital image, but also a number of versions of it in lower resolutions, [1], [10].

The search for the optimal position, as defined by the matching measure in Section II, will be made in a resolution pyramid where each level consists of a representation of the distance image. The search will start at a very low resolution and the results from the low resolution matching will guide the computations at finer levels. In low resolution the computations are fast and noise and irrelevant small features are averaged out. On the other hand the image data are coarse and thus not very accurate. In high resolution the data are accurate, but perhaps noisy, and computations may be slow. Ideally only final adjustments are made in high resolution.

There were several problems in expanding the algorithm to a resolution pyramid. First, care has to be taken in the construction of the resolution pyramid itself, Section III-A. Perhaps the most difficult problem is the computation of suitable step-lengths for the parameters in the image transformation equations. If the steps are too big the final match will be poor. If the steps are to small the algorithm will be slow and will perhaps not converge to the optimum. This step-length problem also occurs in the single-resolution algorithm, and in fact in all parametric

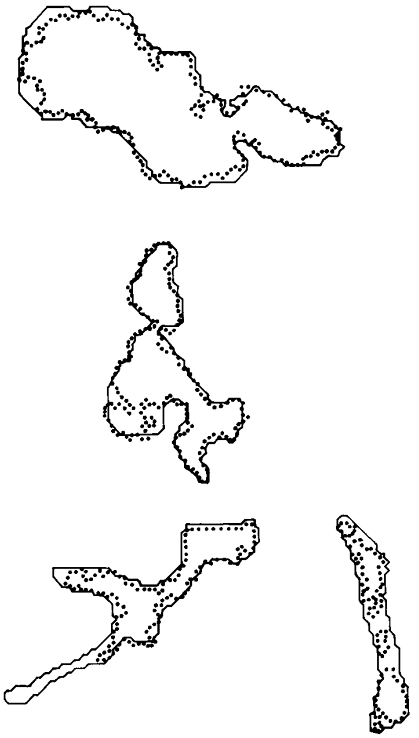

Fig. 4. The optimal positions for some fuzzy lake edges from an aerial photograph (dotted) in the map (solid).

matching algorithms. One solution of this problem is outlined in Section III-C.

As the example in Section II showed, the edge distance function to be minimized at each resolution level is usually nonconvex and have many local minima, apart from the global one. A simple optimization strategy, which finds a close local minimum, is used, Section III-E. If the start positions are too few, or unsuitably placed, the optimal position will not be found. If they are very many the HCMA will become slow. However, the problem of chosing start positions, Section III-D, is not too difficult, as many positions can be investigated quickly in low resolution. The choice of the resolution level where the computations should start is also important.

As the search for the optimal position is started in very many positions, an effective way of rejecting bad positions early (i.e., in low resolution) must be found. If good rejection criteria cannot be found, then imbedding the algorithm into a hierarchical structure is of no real value. This is a key problem, which has no simple solution. The heuristic rejection criteria used here, Section III-F, have been effective in all tested applications.

A. Resolution Pyramids

The most common resolution pyramid is constructed in the following way. The original image is the bottom, zero, level of the pyramid. From this level the next, first, level is created by letting each block of four pixels be represented by one pixel at the next level. The single pixel in the upper level is called the "father" and the four pixels in the lower level are called the "sons." If the image is a gray level image, then the value of father is an average, usually arithmetic or median, of the values of the sons. The new pyramid level is itself used to create a new level. The process is repeated until only one pixel is left.

The image that will be converted into a pyramid in the HCMA is the binary predistance edge image. For binary images the rule used for creating the pyramid is not self-evident, as the average value of the sons must be transformed to zero or one. The only acceptable rule is: if any one of the sons is a one, then the father becomes one. The resulting pyramid is called an OR-pyramid, as the rule is equivalent to the logical operation OR. This rule ensures that all connected edges are connected at all resolutions and that no object, or even isolated point, in the predistance image disappears, [12]. The pyramid created from the edge image is henceforth called the *edge pyramid*. The lowest resolution levels in the pyramid are usually not used, as they are too coarse to contain any useful information.

The chamfer 3-4 DT, Section II-A, is applied to each level of the binary edge pyramid. Thus a pyramid of distance images, a *distance pyramid*, is created.

The distance pyramid could also have been computed in at least two other ways. It could be computed from the bottom level distance image directly, by averaging, without creating the edge pyramid. It could also be computed from the original gray-level image: first the image is converted to a pyramid, then the edges are extracted at each level, and finally the DT is applied. The first of these alternative methods is less computationally complex than the chosen one (no intermediate pyramid), but the other one is more complex (unless the edge extraction algorithm is very simple).

Both these alternatives are unsuitable. Let us call the three constructions "HCMA pyramid," "average pyramid" and "gray-level" pyramid. In the HCMA pyramid it is obvious which fathers have a piece of an edge in any of their great-great. . . grandsons (the zero ones) and which fathers do not (the nonzero ones). In the average pyramid however, the averaging will hide the edges. If the edges are one pixel thick, then there will be *no* zero pixels at the 1st and following levels. This smearing can and will cause difficulties: Suppose that the start position of the polygon is perfect, i.e., the edge distance is zero at the original level of the distance pyramid. In the HCMA pyramid, the edge distance will be zero at all levels. In the average pyramid the edge distance will be greater than zero at all but the 0th level. This is not only unnatural, as the position is correct, but it also makes the matching algorithm inconsistent, i.e., an optimal start position may end in a non-optimal position. Inconsistent algorithms may easily fail. The same problem occurs for the gray-level pyramid: there is no guarantee that the edges will be found at exactly the same positions at all levels and thus no guarantee that the algorithm is consistent.

The edges in the HCMA pyramid does become thicker, and thus blurred, at low resolutions. However, the quality of the low-level results are unimportant, as long as the pyramid construction ensures that the final results, always computed at the best resolution level, are optimal. Convergence and consistence of the matching algorithm will be discussed in Section III-G.

B. *Input Data and Transformation Equations*

The input data to the HCMA are: the prepolygon image; the approximate transformation equation parameter values, (i.e., the approximate position of the polygon) together with an estimate of the accuracy of the approximation; and the predistance image.

Edges are extracted from the predistance image and a distance pyramid is created. If the same predistance image is used repeatedly, then storing the "ready-to-use" distance pyramid is preferable, even though this requires more memory, as it will speed up the algorithm.

Edges are also extracted from the prepolygon image (unless the polygon is a template). In most cases it is neither necessary nor desirable to use all edge points. A suitable application dependent set of edge points is chosen and becomes the polygon. This polygon can consist of a set of isolated points, a few short edge segments, or a continuous contour. If the two images are to be brought into the same coordinate system, then the polygon points should be scattered over the whole image, as the correspondence will probably be best in areas close to polygon points (Section V). If a certain object is searched for in an image, then the polygon should be a template, i.e., an ideal outline of that object, rather than edges extracted from an image, (Section IV).

The polygon origin should be located in a suitable position. A natural choice is the center of the smallest rectangle enclosing the polygon points. Other origins that could be considered are the center of gravitation and the center of the ellipse of inertia. If the polygon covers a large part of the prepolygon image, then the center of this image would probably be the best choice.

The number of parameters in the transformation equations is usually 2 to 6. If the polygon point is (x, y) and the position of that point is (Xr, Yr), then the equations become

$$Xr = \text{PTX}\ (p_1, p_2, \cdots, p_n, x, y),$$

and

$$Yr = \text{PTY}\ (p_1, p_2, \cdots, p_n, x, y), \tag{5}$$

where PTX and PTY are the polygon transformation equations for the X and Y coordinates respectively, and p_i are the parameters. The values of Xr and Yr are not dependent on the current pyramid level. To determine which pixel (X, Y) in the distance image that is hit at each level, Xr and Yr are rounded to the appropriate resolution. At level n the formulas are

$$X = \left\lceil \frac{Xr + 2^n - 1}{2^n} \right\rfloor \text{ and } Y = \left\lceil \frac{Yr + 2^n - 1}{2^n} \right\rfloor. \tag{6}$$

C. *Parameter Step-Lengths*

The choice of parameter step-lengths is a crucial part of the HCMA. A general solution to this different problem is outlined here.

In the optimization algorithm, the position of the polygon is changed by changing the parameter values p_i (5) in discrete steps. To ensure a good match, the polygon position should change very little for each parameter change (as the edge distance function is complex). From the matching point of view, two positions differ if at least one polygon point, (X, Y) in (6), hits a different distance image pixel. The smallest step in parameter space that actually changes the position must be determined. The size of this step is computed for each parameter separately, the other parameters being constant. This strategy is powerful enough for our purposes. The computed value is called *parameter step-length*.

It is easy to determine step-lengths for the translation parameters c_X and c_Y. At the original level of the pyramid the step-lengths are one pixel, i.e., one distance unit. One step in parameter c_X is equivalent to shifting the polygon one pixel down in the distance image. At the next level the pixels are twice as large. Therefore, to change the polygon position, the step-length must be two distance units. At level n in the pyramid the translational step-length is 2^n.

For other parameters, e.g., for rotation, scale, or perspective, the situation is more complex. Translation and rotation change the position of the polygon, but not its size or shape. Scale changes the size and perspective changes both the position, size and shape of the polygon. The step-length of one nontranslation parameter depends on the values of all nontranslation parameters. It also depends the size and shape of the polygon and on the pyramid level. Exact determination of the step-lengths becomes impractical. Thus approximations are used. In most cases the polygon points farthest from the polygon origin will change more than (or as much as) other points when the parameters change. Therefore, the polygon point with the largest distance to the origin will be used in the approximation. This point is denoted (max x, max y).

If f is a differentiable function, and d a small number then

$$f(x + d) \approx f(x) + d^*f'(x),$$
$$\text{i.e., } f(x + d) - f(x) | \approx | d^*f'(x). \tag{7}$$

Insert the PTX-equation (5) in (7). The change in Xr(max x, max y) when the parameter p_i is changed to $p_i + d$ can be approximated by:

$$\left| d * PTX'_{p_i}(p_1, \cdots, p_i, \cdots, p_n, \max x, \max y) \right|. \tag{8}$$

A similar expression approximates the change in PTY.

During the optimization one parameter step is taken both in the positive and the negative direction ($+d$ and $-d$). The position of (max x, max y) will change for at least one of these new positions if the change in Xr or Yr

is larger than $0.6 * 2^n$ at pyramid level n. The smallest step d_i^X in parameter p_i to change $X(\max x, \max y)$ in (6) is thus [using (8)]:

$$d_i^X = \frac{0.6 * 2^n}{\left| \frac{\partial}{\partial p_i} \text{PTX}\,(\boldsymbol{p}, \max x, \max y) \right|}. \tag{9}$$

and the smallest step d_i^Y to change Y is determined by a corresponding expression. The approximate and used step-length for parameter p_i is the minimum of d_i^X and d_i^Y. The factor 0.6 was introduced in the step-length computation as tests in complex applications showed that the matching results are unsatisfactory without it: The steps become too large and thus the final matches are not good enough. The factor 0.6 shrinks the steps to a satisfactory size.

Consider the common case where the polygon transformation is translation and rotation (4). Using our approximation, the step-length for the rotation angle becomes [insert (4) in (9)]:

$$d_{\text{rot}} = \min\left(\frac{0.6 * 2^n}{\left| \sin(\text{rot}) \max x + \cos(\text{rot}) \max y \right|}, \frac{0.6 * 2^n}{\left| \cos(\text{rot}) \max x - \sin(\text{rot}) \max y \right|} \right). \tag{10}$$

Step-lengths for one set of six-parametric transformation equations are discussed in Section V.

In low resolutions the computed step-lengths can become very large. A natural upper limit of the step-length is defined by the start positions. Each start position is only supposed to find the best match in its own small neighborhood in parameter space. Thus, if the computed step-length is larger than the distance between start positions (for each parameter), then the parameter is considered constant.

D. Start Positions

In cases with simple geometry, or if the correct position of the polygon is reasonably well known, the set of start positions can be rather small. If not, a large number of start positions is needed, so that at least one search will find the global minimum.

The best set of start positions is usually a regular grid of points in n-dimensional parameter space. The approximate parameter values define the midpoint of the grid. The accuracy of the approximations is used to determine, for each parameter, intervals that definitely include the optimal values. These intervals define the volume the grid must span.

The start positions must be far enough apart, so that there is a significant difference between them at the coarse resolution start level, but they must still be rather close, as the number of local minima is large. The parameter step-lengths are first computed as described in Section III-C. If the start level is suitable, then a spacing of three a set of scattered points, then for each distance image pixel one of the pixels (e.g., the first) from the shrunk polygon that hits that pixel is chosen. These selection methods do not ensure that all polygon points will hit different pixels, since the polygon is later transformed, but the points will be an adequate representation of the polygon.

Many iterative optimization algorithms for minimizing an n-dimensional function have been suggested. Many of them depend on the function being relatively smooth, so that the gradient can be, if not computed exactly, at least approximated. This is not the case for the edge distance function. Thus an algorithm that does not use the gradient must be used. A simple version of the Gauss-Seidel algorithm was chosen. The same optimization algorithm is used at every pyramid level.

The edge distance is first computed for the start position. Then the parameter step-lengths are computed (Section III-C). The edge distance is then computed for some neighboring positions in parameter space, selected by the rules described below. If a strictly smaller value is found, the parameter values are changed to this new position. or four steps between grid-points will be appropriate in most cases.

What is a "suitable" start level? If the start level is too coarse, then the algorithm may not converge to the optimum, as all start points are too far from it. If the start level is too fine, then the start grid will consist of very many positions and the benefits of the hierarchical approach disappear. As an approximate rule, the objects in the image should still be more or less recognizable at the start resolution, but only just.

Computations at coarse pyramid levels are much faster than at finer levels. The main reason is that the number of points in the polygon is much smaller at coarse resolutions (see Section III-E). Thus fewer look-ups in the distance image and fewer multiplications and additions are necessary to compute each edge distance. A large number of positions can be investigated with reasonable effort. Most of these positions should be rejected early. Ideally only a few should be left at the original pyramid level.

E. Optimization

The distance pyramid, the edge point polygon, and the set of start positions have been computed. Now the actual matching can start. A flowchart for the HCMA is shown in Fig. 5. Each of the boxes will be described below.

When the polygon is superimposed on a low resolution distance image many of the polygon points will hit the same pixels. This will slow down the computations without improving the edge distance. A suitable subset of the polygon points should be used. The point selection is the box "select points" in the flowchart. The polygon is first "shrunk" to the current resolution (6). If the polygon is a continuous contour, e.g., a template, the number of different points is counted and that number of points is selected, evenly spaced along the contour. If the polygon is The process is repeated until no neighbor has a lower edge

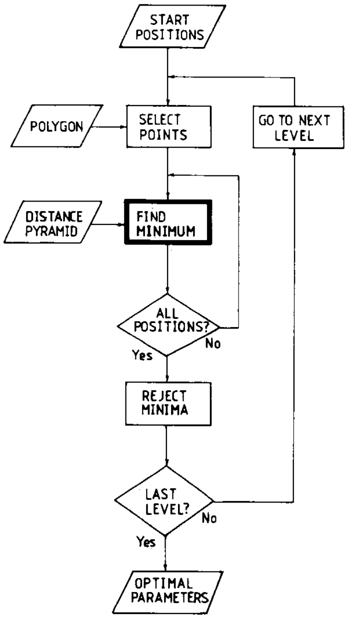

Fig. 5. Flowchart for the HCMA.

distance value, i.e., when a local minimum has been found. This minimization is the central "find minimum" box in the flowchart.

Ideally the function should be evaluated for all neighboring positions. However, in n dimensions each position has $3^n - 1$ neighbors. It is infeasible to evaluate the function in all these positions unless n is very small, e.g., $n < 3$. The function is regarded as being dependent on only one parameter at the time, the others being constant. A step is taken in both directions from the current position and if a better value is found the parameter value is changed. The process is repeated for each of the parameters in a cyclic way. One cycle, going through all parameters, is called one iteration. Only $2*n$ neighboring positions are evaluated in each iteration. The algorithm stops when no change has occurred during one iteration.

For $n = 3$ it is still possible to evaluate all neighbors. Experiments showed that the convergence is faster if all 26 neighbors are evaluated rather than just six (as can be expected). This observation led to a modification of the search algorithm. The polygon is easily transformed by the translation parameters. To find all eight neighbors reached by translation you just add plus or minus one to the midpoint coordinates. Evaluating the function in these positions is thus relatively easy. Each iteration thus starts with checking all eight translational neighbors. Then the neighbors in the other parameters follow, one parameter at a time. Thus $2 * n + 4$ rather than $2 * n$ neighbors are evaluated in each iteration.

If the parameter step-lengths depend on the parameter values it is necessary to recompute them after each iteration. If they do not (e.g., only translation and rotation) it is only necessary to recompute them at each new pyramid level.

The local minima are found for all start positions. The worst local minima are rejected, the box "reject minima" in Fig. 5. The remaining positions are used as start positions at the next pyramid level. In principle some pyramid levels could be excluded, e.g., every other one. Each start position would, in most cases, end in the same position at the final resolution level. However, the main rejection criterion, Section III-F, is dependent on every level being used. The computation continues down to the original resolution level. The smallest remaining minimum is the solution and its parameters are, when the algorithm is successful, the optimal transformation parameters.

F. Rejection Criteria

The success of the HCMA is very dependent on good rejection criteria. Several rejection criteria are used, both looking at each minimum by itself and comparing it to others.

The start grid defines not only the start positions, but also the hypervolume in which the optimum can occur. Local minima outside this hypervolume are rejected. An example: if there is no check on the scale parameter, the transformed polygon could shrink to a single pixel. These obvious restrictions reject very few positions.

Another obvious rejection criterion is the magnitude of the edge distance at the local minimum. If it is too large, that position should be rejected. If the magnitude of the global minimum is known beforehand, then all significantly larger minima can be rejected. However, the magnitude of the global minimum is usually not known beforehand. The main rejection criterion, described below, does not start rejecting minima effectively until after a few pyramid levels have been passed. A large absolute limit, significantly larger than any acceptable optimum, is used. This limit will reject the worst start positions immediately.

For the optimal position, the minimum is of about the same magnitude at all resolutions. The edge distance is somewhat smaller at low resolutions, as the edges are blurred, and thus wider, there. For positions that are far from correct, the situation is quite different. At low resolutions the polygon hits only a few pixels and can thus fit well in many different locations. When the resolution increases the fit rapidly gets worse and worse and the edge distance increases exponentially. To conclude: for good positions the edge distance increases roughly linearly and for bad positions exponentially. This phenomenon is clearly visible in all cases and in all applications tested so far. Therefore a simple criterion is used, designed to test the increase of the edge distance. The first nonzero local minimum is stored for each position. This value, multiplied by a suitable constant, becomes the rejection limit. If the difference between the new minimal value and the minimal value at the previous pyramid level exceeds this limit, then the position is rejected. That "suitable constant" will be called *reject factor*, or *RF*. It is application dependent, but much less so than any absolute limit would be. A small RF will speed up the computations, but risks the rejection of the correct position. A large RF will slow down the computations. The general rule is: the more difficult the matching situation is, the larger RF must be.

The three rejection criteria discussed so far only test one position at a time. This final criterion compares the minima. The edge distance of the smallest minimum rejected by the previous criterion is saved. All larger minima are rejected. In fact, this rejection rule is the one that discards the most minima. The global minimum can thus often be found faster if the search is started in many positions than if it is started only in a few: the more positions, the smaller the smallest rejected minimum will be and the more positions will be rejected by that limit.

One important observation have been made in almost all experiments: what eventually becomes the best positions do not have the lowest edge distances at course resolution levels. There is only a very rough correspondence between the order of positions at coarse levels and at fine levels. Thus any "depth first" search, where only the best minimum (or minima) from the start level is followed through the pyramid, is doomed to fail.

G. *Convergence and Results*

Unfortunately, there is no guarantee that the HCMA will converge to the global minimum of the edge distance function. This function is simply too complex in most realistic cases. However, two propositions hold.

Proposition 1: The HCMA will always converge to some parameter values. It will never be caught in a infinite loop.

Proof: The edge distance function that is minimized is integer valued, as it is the sum of squares of integer distance values from the distance image. The improvement of the function value at each iteration in the optimization can thus never be less than one. The edge distance of the start position is finite, say N. The maximum number of iterations is then $N + 1$. Q.E.D.

Note that the number of iterations does *not* increase with the complexity of the problem. More edge points in the polygon and less accuracy of the estimated position both increase the number of necessary start positions, but not the number of iterations for each position. In fact, using many polygon points would rather decrease the number of iterations, as the global minimum of the edge distance function will become more distinct.

Proposition 2: The algorithm is consistent. If the matching starts in a correct position, i.e., in a position with edge distance zero, then it will also end in that position.

Proof: The edge pyramid is constructed so that the greatgreat. . .grandfather, at any pyramid level, of any edge pixel will also be an edge pixel (Section III-A). If the start position is correct, all polygon points will hit zero-valued pixels in the distance image at all pyramid levels. The edge distance will thus be zero at all levels. No strictly smaller value can be found in any neighboring position. Thus the start position parameters will never be changed. Q.E.D.

If the search is started in a global minimum that is not zero, then the search does not necessarily end in this global minimum. The exact position of the minimum is often not the same at different pyramid levels. Thus the position may drift away during the search. Some tests were made with a six-parametric edge distance function: starting the search at a coarse level in an optimal position showed that the minimum often does shift position. However, in all cases tested the minimum returned to the optimal position at the final resolution.

Unless the edge geometries are very simple, the edge distance function is nonconvex and have many local minima. The longer, the more distinct in shape, the more spread out over the image, the polygon is, the smoother the edge distance function becomes, the sharper its global minimum becomes and the better the HCMA performs. If the minimum is reasonably well-behaved it is probably possible to achieve subpixel accuracy of the optimal position. The edge distance is computed at a number of points around the optimum and an n-dimensional smooth surface is fitted to these values. The minimum in this surface then defines the improved minimal position.

IV. Object Recognition

In this application the HCMA is used to identify and find the location of a number of prespecified objects. The objects are common tools. The implementation of the HCMA used here and in the next section was written in Simula and run on a DEC-10 computer. The program was written to give maximum flexibility and minimum time spent at programming, rather than to be efficient.

A. *The Algorithm*

The camera producing gray-level digital tool images is fixed above a flat surface looking straight down. The lighting has been arranged so that shadows are minimal. Only three parameters are necessary to describe the position of an object: X-translation, Y-translation, and rotation angle (4). Five different tools, scissors, hammer, screwdriver, wrench, and knife were used. They have, on purpose, the same basic elongated form and about the same size. A series of scenes of varying complexity were photographed and digitized. The results were a number of 512 × 512 pixel images with 256 gray-levels. One of the images is shown in Fig. 6.

The edge points in the gray-level image were extracted in the simplest way possible. The image was thresholded at the minimum in the gray-level histogram separating the objects (dark) from the background (light). This minimum was reasonably well defined. Edge points in the binary image were defined by the following rule: an edge point is a black pixel with at least one white 4-neighbor *and* at least one black 4-neighbor that is not an edge pixel (i.e., is in the interior of the black area). The last rule excludes small patches lacking interior points and also short "branches" on the edges. Overlapping objects get one single contour. This drawback is actually exploited, to test the limits of the performance of the algorithm. The

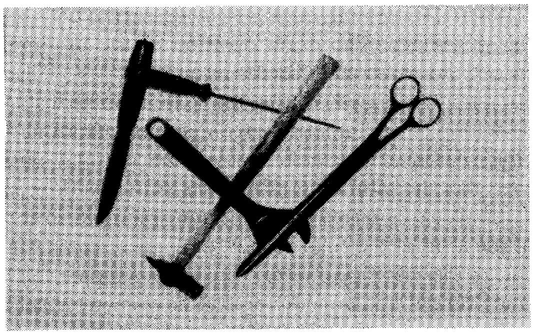

Fig. 6. Tool image to be analyzed.

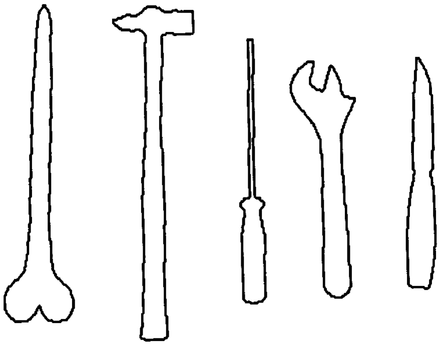

Fig. 7. The template polygons. They have been extracted from a photograph and have then been interactively smoothed.

edge image is first converted into an edge pyramid and then the edge pyramid is converted into a distance pyramid.

In this application there is, strictly speaking, no prepolygon image. The patterns to be matched to the tool image are ideal outer contours of the different tools. These contours are called *templates*. Templates for the five tools were extracted from images, taken with the same camera arrangement as the other photographs. The edges resulting from edge extraction were "cleaned up", i.e., they were smoothed and false edges resulting from highlights were removed. The remaining edge points were listed in polygon files. The polygon origin is the center of the least rectangle enclosing it. The five templates are shown in Fig. 7.

Here the distance image and the polygon image are not interchangeable. In any complex image, with many objects, it is difficult to extract the true contour, or even a part of a true contour, of a single object (there can be shadows, highlights, and overlapping). Thus matching any polygon extracted from the photograph to a "template" distance image may easily fail. When the templates are compared to the photograph edges this difficulty disappears. There is no need to know beforehand which edge points belong to which object. False edges and overlapping will of course disturb the matching, but if a large part of the edge of an object can be extracted, then the object will be found.

The step-length in rotation angle is computed by (10). The smallest step allowed is 0.5 degrees. The search for the optimal position is started in a grid of positions in parameter space. Grids of different densities were tested. The start resolution level in the pyramid was the 4th, 32 × 32 pixel, level. The absolute limit on the edge distance value is set to 10.0. All larger minima are rejected. The most suitable RF value differs for different types of scenes. Some examples are given below.

B. Simple Scenes

First the HCMA was used to determine which tool is depicted in an image showing a single tool, but otherwise equal to Fig. 6. The tool can be any of the five tools placed in any position or orientation. No information on the position was available to the HCMA.

The template polygons consist of a large number of points (up to 600). Every 10th point in the polygon were used, at most. In low resolutions much fewer points are used, as described in Section III-E. The center of the start grid was the center of the image (translation) and rotation angle zero. The grid consisted of 54 positions: 3 × 3 translational points, separated by 30 pixels at the original level, and 6 equidistant rotation angles (i.e., 60 degrees apart).

Each template is matched to the image. The result is either a minimal edge distance or no match (i.e., no remaining position). The image is interpreted as depicting the template with the smallest edge distance optimum. The difference in edge distances between correct and incorrect matches is so pronounced that there is no difficulty in interpreting the images. The optimal edge distances for the different tools when the match is correct are: scissors 0.67, hammer 1.25, screwdriver 0.70, wrench 0.74, and knife 0.71. The optimal edge distances for incorrect matches are more than twice these values. When RF ≥ 2 the correct matches were never rejected, but for RF < 2 the correct positions sometimes were lost. RF should be 2–3 in this case. The HCMA is thus well able to identify an object, by comparing it to a library of possible objects.

The next scene type to be analyzed was images containing many, but not overlapping, tools. There must be more translational start positions, otherwise the algorithm is identical to the single tool case. The different tools in this type of images are easily identified and their positions easily found. The ideal RF value seems to be 4 (a little higher than for single tools), but 3–6 will do nicely.

C. Complex Scenes

If the tools in the image overlap, then parts of their edges will be missing. The extracted edges of the image in Fig. 6 are found in Fig. 8. The corresponding edge pyramid is shown in Fig. 9. The matching is started at the coarsest level in this figure. The original level distance image is shown in Fig. 10.

No estimation of the tool positions is available. The start position grid consists of 120 positions: 4 × 5 translational positions covering the whole image and 6 equidistant rotation angles, as before. Every 10th template

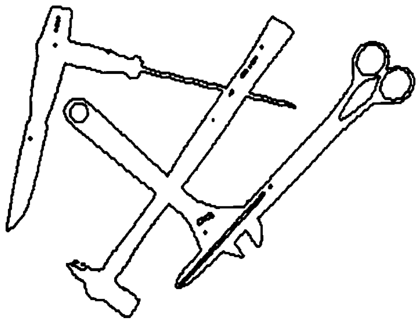

Fig. 8. Edge points from the tool image in Fig. 6.

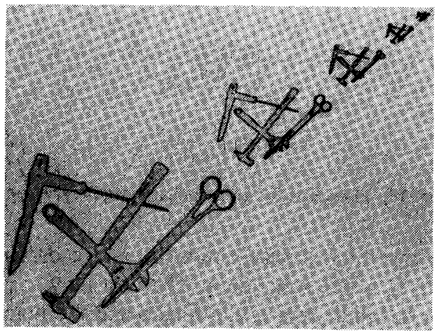

Fig. 9. Edge pyramid computed from the edges in Fig. 8.

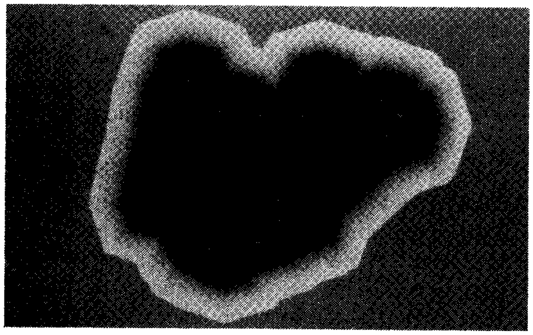

Fig. 10. Distance image representing the original level in the distance pyramid. The distances are gray-level coded: the larger the distance the lighter the tone.

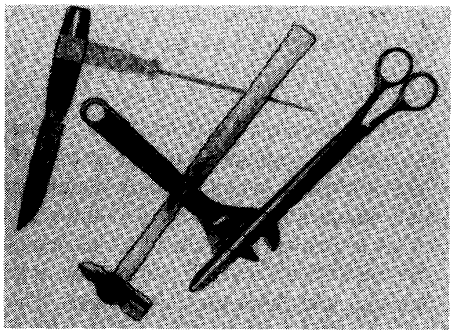

Fig. 11. The optimal positions of the different templates (black) in the image in Fig. 6 (gray).

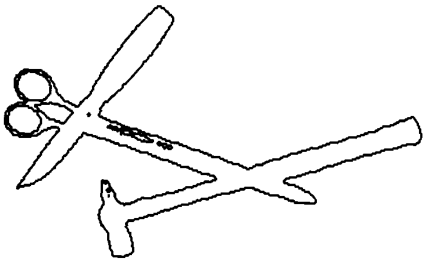

Fig. 12. Tool edge image to be analyzed.

polygon point was used. Two different RF's were tested: 4 and 8. The scissors, hammer, and knife are found without problem in Fig. 8. These are the tools for which the contours are the least disturbed. The wrench is found when RF is large enough, i.e., for RF = 8 but not for RF = 4. The screwdriver is not found (see below). The minimal edge distances are larger than in the previous tests, they are: scissors 1.56, hammer 1.51, wrench 1.42, and knife 1.23. The edge distances increase both because segments of the edge are missing and because the tools no longer lie flat on the surface. The templates for the four successful matchings have been overlayed the original image in their optimal positions, Fig. 11.

The reason that the screwdriver was not found is that the highlight on the scissors create a false edge (see Fig. 8) and this false edge happen to fit the screwdriver very well. The edge distance is there 2.00. The edge of the "true" screwdriver is severly disturbed by the knife, which has obliterated the end of the handle. This, together with the intersection with the hammer, make the edge distance for this second best, but correct, position 2.16. The search fails because the edge of the true tool is severly disturbed and because a false edge happens to create a good false fit.

In the previous image all tools were present. Thus the algorithm have answered the question of *where* they are, but not *if* they are present. That the algorithm can answer this question too is evident from the results in Section IV-B. But what happens for an image with unknown overlapping tools? The edges from such an image are shown in Fig. 12. The tools depicted in the image are easily found, using RF = 4. However, the missing tools are also "found" and the edge distance values of the best false matches are only just a little higher than for the true matches. Thus the edge distance cannot be used to identify false matches here. For RF = 2 the false matches are rejected, but, unfortunately, the true ones are also rejected. This image shows the limit of the applicability of the HCMA: for images with very disturbed edges and doubtful contents, the algorithm is not powerful enough.

One way of overcoming the difficulty with missing contour segments could be to divide the templates into a number of subpolygons. Each subpolygon would be matched to the image. Undisturbed parts of tool edges would fit very well. If several of the subpolygons have good matches in consistent positions, that would indicate the presence of the tool. That one or two subpolygons did not match could be disregarded as due to noise.

D. Conclusions

The results of Sections IV-B and IV-C show that the HCMA can easily recognize an object as one of a predetermined set of objects, when the only disturbance of the edge is some random noise. They also show that finding an object that is known beforehand to be depicted in an image is not too difficult, even if parts of the object's contour are missing and there are false edges. The HCMA seems to be powerful enough to handle disturbed and noisy data at least as well as other recent matching algorithms.

When the fit of the edge can be bad either because the object is missing or because parts of the edge are missing, then the algorithm fails. The measure of correspondence used, being one single number, simply cannot distinguish between these two cases. On the other hand, neither can any matching method that uses a single number as measure of correspondence, and simple preprocessing of the images before matching. For such complex matching problems high-level methods are necessary. Such methods are overviewed in [1].

The CPU-time for matching all five templates, each with about 50 points, 120 start positions, and RF = 4 is about 8 minutes in the present inefficient implementation. Some hypothetical CPU-times have been computed, using data provided by the matching program. If the search were started in all 120 start positions at the original resolution level the CPU-time would be at least 18 minutes. However, then the optimal positions would probably not be found, as the start positions would be too far from the optimum. Consider a grid that has the same number of pixels between the translational start positions as the 120 position grid has at the start level in the hierarchical case and rotation angle steps that are shrunk correspondingly. Then the program would run for 50 CPU-*days*. This estimation gives and indication of how much is gained by using the hierarchical structure.

V. Aerial Image Registration

In this section the HCMA is applied to a demanding problem: transformation of aerial photographs into the map coordinate system. This transformation is called *registration*. When the image has been registered, then objects in the map can be identified in the image and vice versa; changes between different scenes can be easily detected; and images from different sensors (e.g., optical, IR, radar) can be simultaneously displayed and the information from them can be integrated.

A. Problem Definition

Traditionally the registration algorithm consists of three or four parts. A flowchart of the traditional algorithm is found in Fig. 13. It is a combination of the approaches in [7] and [9]. The basic data are a geographical image (in any wavelength band), the approximate position of the sensor with which it was obtained, and a database containing maps of the same area. The best matches between a number of landmarks in the image and the map must be found. The landmarks are usually points, edges, or objects. These must first be identified in and extracted from the image. The approximate sensor position determines an appropriate area from the map database and is also used to hypothesize a first guess of the transformation that puts the features from the image into corresponding map positions. A coarse match between image and map features is made, to get approximate results. Then a fine match is made, to get the best possible matches. The coarse and fine matches often use quite different algorithms. From all the resulting feature matches, that always are somewhat erroneous, the best parameter values for the sensor position is computed. This part of the image registration algorithm is known as the location determination problem, LDP, because the point in space from which the image was obtained (sensor position and orientation) is determined. Last, the image must be resampled (registered) and brought into the map coordinate system.

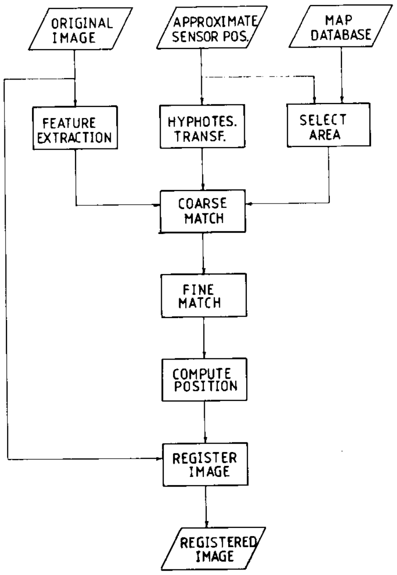

Fig. 13. Flowchart for geographical image registration.

When the HCMA is used, all parts of the registration algorithm between the feature extraction and the registration are merged: the features are matched precisely by varying the sensor parameters. When the match is optimal, the optimal parameters are also known. The coarse and fine matches correspond to different resolution levels in the distance pyramid.

The features used in this example are land/water edges. These edges are clearly visible not only in optical photographs, but also in imagery in many other wavelength bands. Using edges rather than points for matching will give better results, as the edges contain much more information. However, the edges extracted from the photograph will be noisy and differ considerably from the map edges.

The aerial photograph is distorted, not only by translation and rotation, but also in scale and perspective. The polygon transformation equations, in this application usually called the camera model, are six-parametric: translation (two), scale, rotation and perspective (two). The geometry is shown in Fig. 14. The focal length of the camera and the principal point in the image plane (i.e., where the optical axis of the camera pierces it) are known.

The ground coordinate system (X, Y, Z) is denoted GCS and the photograph coordinate system (x, y, z) is denoted PCS. Both have their origins is in the upper left-hand corners of the respective images. The position of the camera is described by the vector $\boldsymbol{c}$, $\boldsymbol{c} = (c_X, c_Y, c_Z)$ in GCS and $\boldsymbol{c} = (0, 0, 0)$ in PCS. The camera is looking along the z-axis in PCS and along the vector s in GCS. The relation between these two vectors is described by three angles: roll, the rotation around the z-axis between GCS and PCS; tilt, the rotation around the y-axis; and pan, the rotation around the x-axis. Any point on the vector s in GCS can be expressed as

$$s = \boldsymbol{c} + kM\boldsymbol{p}, \tag{11}$$

were k is the distance between the PCS origin and the point on s, and M is the 3×3 rotation matrix that transforms PCS into GCS. The elements of M will be given below. The point $\boldsymbol{p} = (x, y, f)$ in the photograph, where f is the focal length of the camera, corresponds to a point $\boldsymbol{g} = (X, Y, 0)$ on the ground (i.e., in the map). The point $\boldsymbol{g}$ is thus the point on the vector s with Z-coordinate zero. The equation for the Z-coordinate in (11) becomes

$$0 = c_Z + k_g(M_{31}x + M_{32}y + M_{33}f). \tag{12}$$

The value of k_g can be solved from (12). Substituting this value into (11) yields these expressions for $\boldsymbol{g} = (X, Y, 0)$:

$$X = c_X - c_Z \frac{M_{11}x + M_{12}y + M_{13}f}{M_{31}x + M_{32}y + M_{33}f}$$

and

$$Y = c_Y - c_Z \frac{M_{21}x + M_{22}y + M_{23}f}{M_{31}x + M_{32}y + M_{33}f} \tag{13}$$

Equations (13) express the relation between map coordinates (X, Y) and photograph coordinates (x, y) and are thus the polygon transformation equations to be used in the matching algorithm, cf (5).

The terrain elevation is not taken into account in this model. This is a reasonable approximation if the ground is relatively flat, or the camera sufficiently high above the ground, or if the whole edge that is matched is on the same elevation. The last condition is fulfilled for lake edges. If the ground cannot be approximated by a flat surface the model becomes more complex. The point on the ground then becomes $\boldsymbol{g} = (X, Y, H(X, Y))$, where $H(X, Y)$ is the elevation at that point. The left-hand side of (12) becomes $H(X, Y)$ and thus k becomes dependent on X and Y. Equations (13) become nonlinear in X and Y (unless H is a very simple function). There is no reason why this more complex camera model could not be used in the HCMA, if necessary.

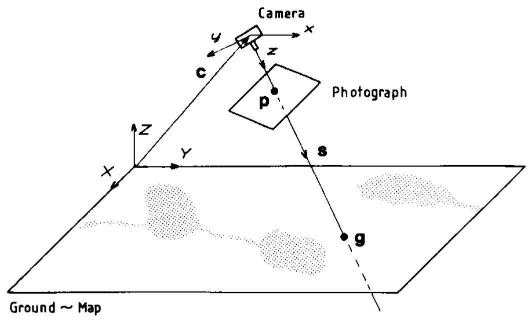

Fig. 14. The camera geometry. The point p in the photograph corresponds to the point g on the ground (map).

To complete the camera model (13) M must be expressed in terms of the roll, tilt, and pan angles. The elements of M become:

$$\begin{aligned}
M_{11} &= \cos(\text{roll})\cos(\text{tilt}),\\
M_{12} &= \cos(\text{roll})\sin(\text{tilt})\sin(\text{pan})\\
&\quad - \sin(\text{roll})\cos(\text{pan}),\\
M_{13} &= \cos(\text{roll})\sin(\text{tilt})\cos(\text{pan})\\
&\quad + \sin(\text{roll})\sin(\text{pan}),\\
M_{21} &= \sin(\text{roll})\cos(\text{tilt}),\\
M_{22} &= \sin(\text{roll})\sin(\text{tilt})\sin(\text{pan})\\
&\quad + \cos(\text{roll})\cos(\text{pan}),\\
M_{23} &= \sin(\text{roll})\sin(\text{tilt})\cos(\text{pan})\\
&\quad - \cos(\text{roll})\sin(\text{pan}),\\
M_{31} &= \sin(\text{tilt}),\\
M_{32} &= -\cos(\text{tilt})\sin(\text{pan}),\\
M_{33} &= -\cos(\text{tilt})\cos(\text{pan}).
\end{aligned} \tag{14}$$

The polygon transformations equations can be parameterized in at least one other way: The angles tilt and pan can be substituted by the coordinates of the point where the optical axis of the camera pierces the round, Fig. 14.

B. Input Data

The map is a digitized version of the land/water overlay of the topographical map of Sweden. The edges were extracted from this binary image, using the rule in Section IV-A. The resulting edge map is the predistance image, to which the photograph edges will be matched. Edge maps can be stored very efficiently, not as images, but as polygons, e.g., in chain coded form [8]. Thus a large number of maps can be preprocessed and stored in a map database. The edge image used here is 512 × 512 pixels, representing 6.4 × 6.4 km on the ground. The edge maps are converted to edge pyramids and then to distance pyramids.

The aerial photographs are the prepolygon images. Several photographs have been registered, to test the HCMA. One of them, Fig. 15, will illustrate the algorithm here. The digitized photograph is 200 × 200 pixels, roughly 2.5 × 2.5 km on the ground. The lakes are the darkest objects in the images. Reasonable lake edges can be extracted by thresholding the gray-level photograph at a carefully chosen level and using the rule in Section IV-A. The edge points found in Fig. 15 are shown in Fig. 16. The edges of the large lakes are extracted as polygons, one for each lake.

C. The Algorithm

The edge polygon is transformed by the camera equations (13). The interdependence of the parameters is a difficulty: a small change in tilt often gives exactly the same result as a small translation in the X-direction after rounding (6). Similarly, a small change in pan angle is often indistinguishable from a small translation in the Y-direction. The *shape* of the polygon will change only if the step in the angle is considerably larger than the smallest step that shifts the polygon. A change in any of the nontranslational parameters can be regarded as a change in shape plus a translation of the polygon origin. These two effects should be separated. Let the results from the original transformation (13) be denoted (X, Y). Let (x, y) be the polygon point, p_i any of the nontranslation parameters, and d_i a step in that parameter. The position of the polygon point in the map where the position of the polygon origin in the map is unchanged (Xc, Yc) is defined as:

$$\begin{aligned} Xc(x, y, p + d_i) &= X(x, y, p + d_i) - \big(X(0, 0, p + d_i) - X(0, 0, p)\big), \\ Yc(x, y, p + d_i) &= Y(x, y, p + d_i) - \big(Y(0, 0, p + d_i) - Y(0, 0, p)\big). \end{aligned} \tag{15}$$

The smallest step-lengths that change (Xc, Yc) are approximated, using (9). The derivation of the actual approximation formulas is straightforward, even though they do become rather complex. The nontranslational parameters now change the polygon shape and/or orientation, but not its location. If the camera equations are unmodified, the step-lengths would become too small: the polygon would almost never change shape. This compensation for the translation of the polygon origin can be seen as an orthogonalization of the parameter space: each parameter should have a unique effect, if possible.

Upper limits on allowed step-lengths are defined by the start grid. If the step-length is larger than the distance between start grid positions, then that parameter is considered constant. A lower limit is also used. For each of the three angles this smallest step is 1.0 degrees. There are thus at most 360 different points in parameter space for each angle. The final matches would have been even better with a smaller value, but the results are good enough for demonstration purposes.

Fig. 15. Original aerial photograph.

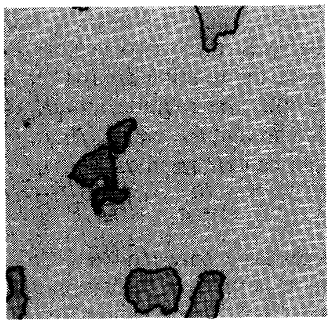

Fig. 16. The lake edges (black) extracted from Fig. 15.

The optimization strategy was described in Section III-E. Thus, 16 neighbors to the current position is evaluated at each iteration. It is very difficult to guess beforehand the magnitude of the optimal edge distance. Therefore no absolute reject limit is used. The search must be started in very many positions, to cover variation in all six parameters, therefore the value of RF is more important here than in the previous application.

D. Sample Results

Many different combinations of input data were tested: different number of polygon points, different choices of polygon segments from the photographs, different start levels, different start grids, and different RF's. Only a few results are presented here, but they are representative of a large number of tests.

One lake is situated roughly in the center of the photograph in Fig. 15. The edge of this lake will first be matched to the map. To get a good overall fit, however, it is necessary to use edge points from different parts of the image. Therefore, two additional lake edge segments were used. One is the southern end of the lake cut off by the upper edge of the scene and the other is the upper half of the round lake at the bottom of the scene. A polygon consisting of these three edge segments is also matched to the map. The total number of edge points is 108 for the single lake and 337 for the three lakes. The polygon points used were every fifth edge point, unless otherwise stated (i.e., 41 and 67 points).

The start positions are a grid roughly around the "true" parameter values, known from the approximate camera position. Five different c_X and c_Y values, forming a 5 × 5 grid covering the whole 512 × 512 map, and three different widely separated height, roll, tilt and pan values, in all 2025 start positions, are used. The start positions were close enough so that several of them found the optimal position. The best start level in the pyramid was the fourth (32 × 32) level. Surprisingly enough even the tiny fifth (16 × 16) level gave acceptable results in most cases.

For each test run the number of minima left at each pyramid level was recorded. This number should be as small as possible, but still the global minimum must not be rejected. The optimal edge distance value was also recorded.

First different RF's were tested. For RF = 1.0 the results are not acceptable: in many cases all minima are rejected. For RF = 1.5 minima remain in all cases, but in some cases the optimal edge distance is slightly worse than the optimum for RF = 2.0. However, for RF = 2.0 there were many false minima left at the final pyramid level. For the three-lake polygon with RF = 1.5 there remained 1650 (of 2025) minima at the fourth level, 566 at the third, 25 at the second, 7 at the first, and 2 at the final level. With RF = 2.0 the same number of minima remained at the fourth level (the rejections start at the second level used), 862 remained at the third level, 54 at the second, 16 at the first, and 10 at the final level. The optimal edge distance was 0.79, which is a very good fit. For the central lake polygon the optimal edge distance was 0.67. In this application RF should be higher than 1.5, but, considering the many false minima that remain, not necessarily as high as 2.0. For RF = 2.0 the CPU-time used is about 30 percent longer than for RF = 1.5.

The same matchings were also made using every second edge point in the polygon, i.e., 104 and 168 points. This change will only affect fine resolutions, because in low resolutions much fewer points are used in both cases. Differences begin to occur at the 2nd (128 × 128) pyramid level. The final results are only partly comparable. The same parameter values will give slightly different edge distances, because different edge points are used. Using more polygon points (2.5 times as many), consumes more CPU-time, but, as the differences occur only at the fine levels, the CPU-time increased only 20-35 percent in the tested cases. There seemed to be no discernible difference in the results between using every fifth or every second edge point. Every fifth pixel defines the lake edges well enough.

The optimal camera parameters have now been computed, using one or three lakes. Now the photograph will be registered, to see how good these parameters are. Finding good registration algorithms is not a trivial problem. Here a very simple one is used, as the registration of the photograph is made only to check the results of the HCMA. First the size of the registered photograph is determined. The camera equations (13) transform straight lines into straight lines. Thus the registered positions of the corners of the photograph will define the area covered by the registered image. These positions are easily computed, using (13). An image of the correct size is defined. For each pixel (X, Y) in the new image, the point in the photograph corresponding to the center of that pixel is computed, using the reverse camera equations. The center of (X, Y) will fall into some photograph pixel (x, y). The value of (X, Y) is set to the value of (x, y). This strategy is called the "nearest neighbor" algorithm. The reverse camera equations, that transform points on the ground into points in the photograph, can be derived from (13). They became (in the notation of Section V-A):

$$x = f * \frac{N_{11}(X - c_X) + N_{12}(Y - c_Y) - N_{13}c_Z}{N_{31}(X - c_X) + N_{32}(Y - c_Y) - N_{33}c_Z}$$

and

$$y = f * \frac{N_{21}(X - c_X) + N_{22}(Y - c_Y) - N_{23}c_Z}{N_{31}(X - c_X) + N_{32}(Y - c_Y) - N_{33}c_Z}, \quad (16)$$

where N is the inverse of the matrix M. As M is orthonormal, N is simply the transpose of M, $N_{ij} = M_{ji}$.

The photograph in Fig. 15 was registered using the optimal camera parameters, found when using the three lake polygon. The registered image should now be in map coordinates. To check this, the lake edges from the map were overlayed the registered image. The resulting composite image is shown in Fig. 17. If the registration is successful, the map lake edges should fit the photograph lake edges well. The fit in Fig. 17 is good. When only the central lake was used the fit became far from good: The map edge was up to five pixels from the photograph edge in the parts of the image farthest from the central lake.

E. Conclusions

This aerial image registration application is the most complex one of those where the HCMA has been tested. The main reason is that the search for the optimal match is six-parametric, which means that very many points in parameter space must be evaluated. The various variables in the HCMA, e.g., start level, start positions, choice of polygon points, reject factor, and parameter step-lengths, must be rather finely adjusted if the algorithm is to be successful. If this adjustment is well done, and if edge points from all parts of the photograph are used, then the HCMA solves the Location Determination Problem adequately. The camera position becomes known well enough, so that the edges in the registered photograph and in the map correspond well.

The original chamfer matching algorithm [2] was used for this application: matching edges from an aerial photograph to a map. A coastline was extracted from a map

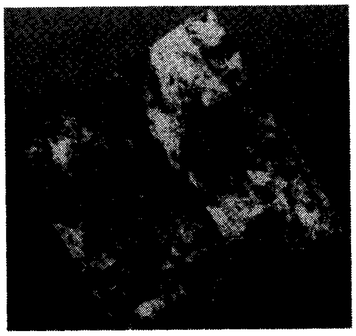

Fig. 17. The photograph, Fig. 15, registered according to the camera position found matching three different lake edges. Lake edges from the map are overlayed in white.

and compared to a coastline from an aerial photograph. The map was there the prepolygon image and the photograph the predistance image, i.e., the opposite of the assignations used here. That choice of prepolygon and predistance images has two serious disadvantages. The worst is that the same edge(s) must be identified in both images *before* the matching starts, as the correct edge must be chosen from the map. Secondly, if a new photograph, or another edge from the same photograph, is to be matched, then a new distance pyramid must be computed from the photograph edge(s), and a new polygon must be extracted from the map. These disadvantages are not quite obvious in the original work, as there was only one long continuous contour present in both the map and the photograph, and as the matching was done only at one resolution.

This algorithm for six-parametric matching can also, with few changes, be used for other applications. It is especially suited to stereo matching, since extra or missing edges (due to different perspectives) do not disturb the HCMA very much.

VI. Conclusion

The first step in the development of the hierarchical chamfer matching method was to improve the matching measure, i.e., the edge distance. This improvement was achieved by using the root mean square average instead of the arithmetic one and using a distance transformation that is a better approximation of the Euclidean distance than the original one. The second, more difficult and more awarding, improvement of the algorithm was to imbed it into a hierarchical structure.

There were several problems with expanding the algorithm to function in the resolution pyramid. The construction of the distance pyramid itself must be appropriate, otherwise the algorithm will not be consistent and will thus converge badly, or not at all. The choice of start resolution level and of the grid of start positions is not very critical, but still not unimportant. The rejection criteria used, that stops further computation from bad positions, must be good enough, so that few computations are done in high resolutions. The heuristic criteria developed here seem to be reasonably efficient in all the different applications were they have been used. The most difficult task is to compute suitable step-lengths for the transformation equation parameters in the optimization algorithm. If the step-lengths are not within quite narrow ranges, then the algorithm will not give acceptable results. The solution used to compute suitable step-lengths can not be called elegant, but it does seem adequate. The size of the smallest steps allowed for each parameter is also important: small steps ensure a good final match. However, very small steps slow down the computations (the optimum is then farther away from a random start position).

The HCMA is rather insensitive to random noise. The edges can be several pixels away from each other, on the average, as long as the differences are nonsystematic. Missing edge segments, false edges, and slight geometrical distortions can also be tolerated. The main reason for this noise insensitivity is that the matching measure is an average of the individual fit of many different edge points. The tool matching application shows the limits of the algorithm: the HCMA can easily recognize an object as one of a set of predetermined objects. It can also find an object that is known beforehand to be depicted in an image, even if segments of its edge are missing and there are false edges. However, when the match can be bad *either* because the object is missing *or* because segments of the edge is missing, then the algorithm often fails. The matching measure, being one single number, cannot distinguish between these two cases.

The arithmetic operations in the HCMA are not very complex. However, as is always the case in image processing, large volumes of data have to be handled. In fact, in the implementation used here a large proportion of the CPU-time is used for shuffling pixel values. An efficient implementation should use special hardware capable of performing parallel operations and capable of handling image data efficiently. The computation of the distance images can be made in parallel. So can the transformation of the polygon points and the now time-consuming computation of the edge distance. The computation of each edge distance would need ony one look-up in the distance image. The trigonometric functions needed for the polygon transformation equations should be listed in look-up tables, as they are needed only for a limited number of values (there is a smallest step-length in angle parameters). Using special equipment, there is no reason why an implementation of the HCMA could not be very fast.

In what ways can the HCMA be further developed? Implementing it on parallel hardware, as suggested above, is one obvious improvement. The function to be optimized is very irregular, except for very simple edge configurations. Therefore, using more sophisticated optimization algorithms would probably bring only marginal improvement. Some modifications of the optimization algorithm have been tested, without much success. Two better ways to improve the algorithm are probably to use better start positions (perhaps found by additional preprocessing) and better rejection criteria. The HCMA can also be combined with other clues to find correct matches faster, e.g., combining it with other matching measures or using image context to discriminate between positions. An example of the latter in the geographical image registration application would be to ensure that the water is on the right side of the lake edge.

For many parts of the HCMA a formal analysis of the algorithm is difficult or even impossible. The solutions of the problems that occured when the algorithm was developed are often heuristic. However, the HCMA has been tested in several difficult and quite different applications. The interested reader can find a presentation of results of many experiments with the HCMA elsewhere, [5]. The results of these tests are good, even surprisingly good. The correct matches *are* found, with an acceptable amount of computational resources. Thus the HCMA is an excellent tool for edge matching, as long as it is used for matching tasks within its capability.

Acknowledgment

The author wishes to thank Prof. G. Dahlquist, Prof. J.-O. Eklundh, Dr. T. Elfving, and Dr. S. Nyberg for their help, in various ways, in creating this algorithm.

References

[1] D. H. Ballard and C. M. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982, pp. 106-111, 352-382.

[2] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *Proc. 5th Int. Joint Conf. Artificial Intelligence*, Cambridge, MA, 1977, pp. 659-663.

[3] G. Borgefors, "An improved version of the chamfer matching algorithm," in *7th Int. Conf. Pattern Recognition*, Montreal, P.Q., Canada, 1984, pp. 1175-1177.

[4] —, "Distance transformations in arbitrary dimensions," *Comput. Vision, Graphics, Image Processing*, vol. 27, pp. 321-345, 1984.

[5] —, "On hierarchical edge matching in digital images using distance transformations," Dep. Numerical Analysis and Computing Science, Royal Inst. Technol., Stockholm, Sweden, Rep. TRITA-NA-8602, 1986.

[6] —, "Distance transformations in digital images," *Comput. Vision, Graphics, Image Processing*, vol. 34, pp. 344-371, 1986.

[7] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381-395, 1981.

[8] H. Freeman, "Computer processing of line drawing images," *ACM Comput. Surveys*, vol. 6, 1974.

[9] J. G. Hardy and A. T. Zavodny, "Automatic reconnaissance-based target-coordinate determinations," in *SPIE Conf. Proc.*, vol. 281, 1981, pp. 95-104.

[10] A. Rosenfeld, "Multiresolution image representation," in *Digital Image Analysis*, S. Levialdi, Ed. London: Pitman, 1984, pp. 18-28.

[11] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, vol. 2. New York: Academic, 1982, pp. 205-219.

[12] S. L. Tanimoto, "A hierarchical cellular logic for pyramid computers," *J. Parallel Distributed Comput.*, vol. 1, pp. 105-132, 1984.

Gunilla Borgefors received the M.S. and "Tekn. lic." degrees in applied mathematics from the Linköping University of Technology in 1975 and 1983, respectively. The thesis presented for the Tekn. lic. degree treated statistical models for the estimation of maintenance times from small samples. She received the Ph.D. degree in numerical analysis from the Royal Institute of Technology, Stockholm, Sweden, in 1986. The subjects of her dissertation were hierarchical edge matching and digital distance transformations.

She was employed by the Department of Applied Mathematics at the Linköping University of Technology during 1975-1981. Since 1982 she has been employed as a Senior Scientist at the Swedish Defence Research Institute, Department of Information Technology, also in Linköping. Her current interests are computer vision and discrete geometry.