A Pyramidal Approach to Convex Hull and Filling Algorithms

M.G. Albanesi, M. Ferretti, L. Zangrandi
Dipartimento di Informatica e Sistemistica, University of Pavia,
Via Abbiategrasso 209, I-27100 Italy

Abstract In the paper, a class of algorithms for filling concavities in binary images is presented. The paradigm of computation is a serial, multi-resolution approach. Among the algorithms which have been implemented, the most significant one is fully described, while for the others, some hints are given on the computational complexity. The algorithm for the approximation of the convex hull has a complexity which is linear in the image dimensions, provided that the multi-resolution is already available. The performance of the algorithms has been measured on a broad set of images and experimental results are reported. Final considerations about parallelization are also given.

1 Preliminary considerations

In binary images, the problem of describing objects often is formulated in terms of an efficient characterization of silhouettes, which are identified by their convex contour. Therefore, in the analysis of the image, the determination of the *convex hull* of the shape or the filling of concavities are used in many applications. It is useful, for the following discussion, to distinguish between two classes of approach; *planar* and *hierarchical*. In the first class, the data structure involved in the computation is a bi-dimensional array, in which operations may occur *locally* or *globally*. On the contrary, in the hierarchical approach, a multi-resolution version of the input image is built, and the computation occurring on a generic level of resolution usually exploits information provided by other levels.
Planar versions of algorithms for the computation of filling have been proposed [2] [3], in which computation occurs locally and iteratively in a 3 x 3 neighbourhood. A hierarchical approach [4] applies iteratively the planar algorithm [3] in each level, followed by a projection operation to pass to the higher resolution level.
The new methods for convex hull and filling here proposed belong to the serial paradigm of computation, performed in multi-resolution. In particular, three algorithms have been implemented for the filling task (named *Fill1*, *Fill2* and *Fill3*, respectively), and in the following section the one with best performance (*Fill3*) is described carefully. For the other two (*Fill1* and *Fill2*), which have been preliminary in our study of hierarchical approaches, we will give some hints on their computation complexity.

2 Hierarchical approaches to convex hull and filling computation

The class of the three algorithms is characterized by the computation of the OR-pyramid on the input binary image; the first algorithm (*Fill1*) applies the classical planar method [2] on a generic level k (where k is an input parameter of the procedure). The computation is followed by a set of projections on the lower levels, until the base is reached. Projection is associated to a refinement operation, which is necessary to maintain the convexity of the shapes.
We have computed [5] the computational complexity of the algorithm, which is of the order $O(N^2)$, for an image of NxN pixels.
This algorithm gives an approximated version of the filling; the error, defined as the number of pixels beyond the boundary of the filled concavity due to the project operation, can be evaluated [5] as $2^{k+1} / \sqrt{2}$. In order to remove this error, a second algorithm can be proposed (*Fill2*), in which the refinement is performed directly on the basis of the pyramid at the end of the projection phase.
In order to understand the strategy of the third algorithm for filling (*Fill3*), it is necessary to introduce a new preliminary algorithm; it aims at finding a convex hull approximation of an object [6] to be used as a starting point in the strategy for filling.
Let us consider a digital image $\Im$ defined on N x N pixels. The approximation of the convex hull is done by considering n straight lines which are tangent to the edges of the object and which envelope the internal concavity of the object. The approximation is done under the following assumptions:

1) The image is binary and an object is defined as the set of black pixels which are contiguous, according to a specified topology (see point two).
2) The algorithm uses the 8-connected topology.
3) The algorithm produces one convex hull approximation for the whole image (see figure 1); therefore, if more than one object are present, the convex hull is the one which includes all of them.
4) The algorithm can approximate the convex hull in n directions ($n \geq 3$); in the following, only the 8 directions are considered (0°, ±45°, ±90°, ±135°, 180°). This is not a strict limitation, because this choice is meaningful to show the effectiveness of the algorithm in most of the considered cases (see section 3).

The algorithm for the convex hull (*CH1*) is formulated in a multi-resolution frame, and it can be described by the following procedure:
Algorithm CH1:
Step 1: On the given image $\Im$, an OR-Pyramid is built up to the 2x2 level. It consists of $\log_2 N$ levels ($l = 0, 1, \ldots \log_2 N - 1$). Level $l = 0$ is the original image $\Im$ level $l = \log_2 N - 1$ is a 2 x 2 pixel image.
Step 2: Starting from level $l = \log_2 N - 1$, n searches are performed, one for each direction D_j (j = 0, ±45, ±90, ±135, 180) of the straight lines which approximate the convex hull. In our case, 8 searches are performed in a serial way. The search consists of finding the first straight line at a given direction D_j which crosses the

object, namely its tangent at that direction D_j. This assumes that the image is scanned along a direction which is normal to D_j. When the line is found, the tangency point is marked as P_j and its coordinate (x_j, y_j) are stored. The step ends when all the 8 lines and the corresponding tangency points are determined.

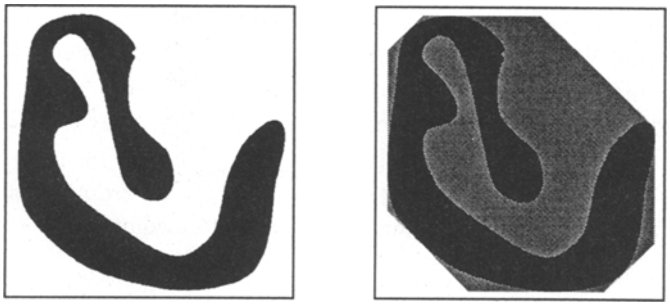

Figure 1. An example of the convex hull approximation: original image (left) and result (right). In the result, the concavity has also been filled.

Step 3: Step 2 is repeated at level l-1, but the search of the 8 lines is spatially restricted by the results of the search at the previous level. In fact, for a given direction D_j the scan of the image at level l-1 starts from the line which passes through the coordinates $(2x_j, 2y_j)$

Step 4: Step 3 is iterated until level $l = 0$ is reached. The result in level $l = 0$ is the output of the algorithm, namely the coordinates of eight points in the image: each point is associated to one direction D_j, and the edges of the convex hull are approximated by the eight straight lines which pass through the resulting points.

The advantage of using a multi-resolution approach is the possibility of restricting the search at a generic level l by using information (i.e. the coordinates of the tangency points) provided by the algorithm at level l-1. This gain can be quantitatively evaluated in the following way: a digital straight line L_j represented by a couple (D_j, P_j), namely passing through point P_j with direction D_j, computed at level l has obviously a thickness of one pixel. The search at level l-1 is restricted to a set of lines, which results from the project of line L_j. The number of lines of this set is 2, because of the topology of an OR-pyramid. This means that the search for each level and each direction requires at most 2 trials. Each trial requires at most $\sqrt{2}N / 2^l$ computations, because there is one possible computation of a new coordinate pair for each pixel of the corresponding straight line of the image at resolution level l. For this reason, the whole complexity of the algorithm can be evaluated [5] as of *O(n N);* the cost of convex hull approximation is linear with respect to the image dimension N. This overcomes the square complexity $O(N^2)$ of a planar approach. Obviously, in a serial computation one should consider also the cost of the construction of the OR-pyramid (Step 1). For this reason, the whole complexity of *CH1* is asymptotically $O(N^2)$. However, software implementation of

the algorithm shows that the execution times are preferable in the hierarchical approach, if compared to the planar one, due to smaller constants (see section 3).
The algorithm *CH1* can be modified into algorithm *Fill3* in order to produce the filling of the concavities of the object (see figure 1):
Algorithm Fill3:
Step 1: On the given image $\Im$, an OR-Pyramid is built as in Step 1 of *CH1*. A second pyramid Ψ of the same dimensions is created and it is filled with ones. This second pyramid will contain the result of the filling in its level $l = 0$.
Step 2: On a generic level l, starting form $l = \log_2 N - 1$ Step 2 of *CH1* is performed, with a little overhead: it consists of filling with zeros the pyramid Ψ in every pixel belonging to each straight line which does not cross the object. Before executing step 3 of *CH1*, a projection of zeros is performed on level l - 1 in pyramid Ψ .
Step 3: Step 2 is iterated until $l = 0$. Pyramid Ψ contains the result of filling.

The overhead due to the building of pyramid Ψ, essentially in the projection phase, raises the algorithm asymptotic complexity to $O(N^2)$, comparable to the planar approach. Experiments show that the execution times are preferable in the hierarchical approach, even if the gain is not high (see next section for details).

3 Experimental Results

This section analyzes the results of software implementation of the hierarchical approaches *Fill1*, *Fill2* and *Fill3*. The test has been done on a series of binary images, realized by thresholding naturalistic gray level images (see figures 2a-2b).

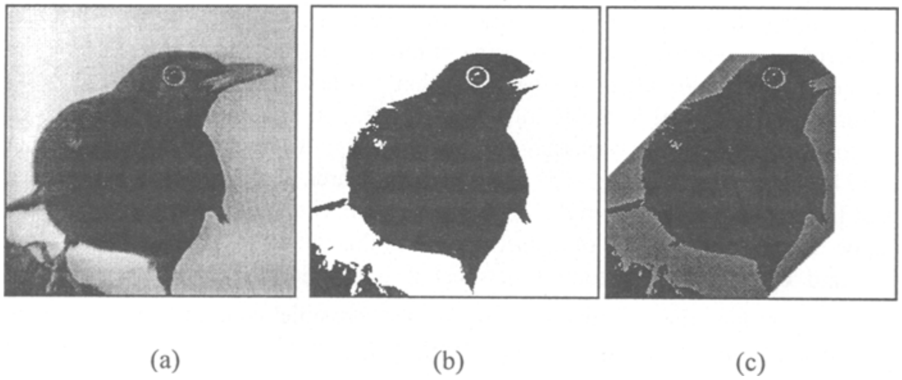

Figure 2. An instance of one of the test:: (a) gray level image "blackbird", (b) binarized image, (c) the result of filling algorithm *Fill3* on image "blackbird".

In the test, CPU times have been measured; the algorithm have been implemented in C++ on a HP 9000/735 workstation. In figure 3 a plot of the CPU time, expressed in seconds as a function of the image linear dimension N, is shown for a comparison

between algorithms *Fill1* and *Fill2*, namely the first two hierarchical algorithms. CPU time has been computed running the algorithms on synthetic images of different size. Experiments confirm the theoretic evaluation of time complexities: time growth is quadratic, but the constants for algorithm *Fill1* are smaller.
Figure 4 shows a plot of the CPU time, expressed in seconds as a function of the image linear dimension N, for algorithm *Fill3*, compared with a corresponding planar serial version (*PL*). In this version, the simple search for the eight straight lines which envelope the concavity is performed at the maximum resolution (N). Experiments confirm the validity of algorithm *Fill3*, expecially for higher image dimensions (N > 800) with a maximum gain of about 30% (N = 1600). Moreover, algorithm *Fill3* is preferable if compared with *Fill1* and *Fill2*: this can be easily verified by comparing plots of figure 3 and 4. For example, for an image dimension of 1024 x 1024, *Fill1* and *Fill2* take over 18 and 20 secs, respectively, while *Fill3* takes 3.4 secs, with a maximum gain of 83%.

4 Conclusions: towards parallelization

In the paper, a hierarchical approach to filling concavities in binary images is introduced. The proposed algorithms are serial, but few consideration can be done for what concerns parallelization. In a parallel pyramidal machine, the problem of filling objects can be solved in constant time if an operation of propagation is available in hardware ([1],[7]).
In a standard SIMD pyramid of $\log_2 N + 1$ levels, the n equations defining the approximation of the convex hull can be obtained in $n\log_2 N$ steps. In fact, the algorithm considers in turn each direction D_j ($1 \leq j \leq n$): each processor computes its distance from the line oriented along D_j and passing through one of the four corners of the basis. The minimum among such distances is associated to the point of tangency or direction D_j; such minimum is extracted using the pyramid by well known algorithm [6] in $\log_2 N$ steps. At this point, the n lines of the approximation of the convex hull are detected, so the filling can be built in a constant time.

References

1. V. Cantoni, V. Di Gesù, M. Ferretti, S. Levialdi, R. Negrini, R: Stefanelli: The Papia System, Journal of VLSI Signal Processing, 2, 1991, pp. 195-217.
2. J. Sklansky, L. Cordella, S. Levialdi: Parallel detection of concavities in cellular blobs, IEEE Trans. Computers, Vol. C-25, Feb. 1976, pp. 187-196.
3. G. Borgefors, G. S. di Baja: Filling and analysing concavities of digital patterns parallelwise, Visual Form - Analysis and Recognition, Plenum ed., 1994, pp. 57-66.
4. G. Borgefors, G. S. di Baja: Methods for hierarchical analysis of concavities, Proc. Int. Conf. 11th IAPR, Vol C, 1992, pp. 171-175.
5. L. Zangrandi: Strategie gerarchiche di raffinamento in algoritmi per l'elaborazione di immagini, Thesis, Dipartimento di Informatica e Sistemistica, Università di Pavia, 1994.

6. R. Klette: On the Approximation of Convex Hulls of finite Grid Point Set, PRL; vol. 2, 1983, pp. 19-22.
7. M. J. Duff: Propagation in cellular logic arrays, Proc. Workshop on Picture Data Description and Management, 1980, pp. 259-262

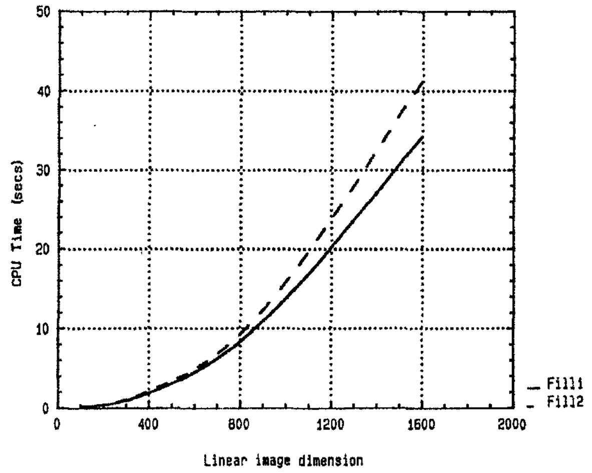

Figure 3. CPU time, expressed in seconds, as a function of the image linear dimension N, for a comparison between algorithms *Fill1* and *Fill2*.

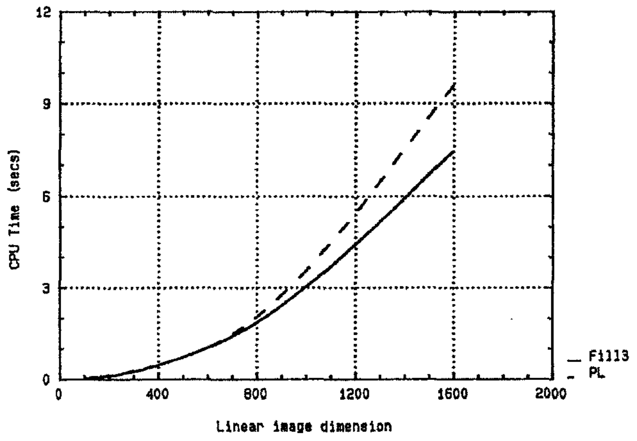

Figure 4. CPU time, expressed in seconds as a function of the image linear dimension N, for algorithm *Fill3* and its corresponding planar version (*PL*).