2D Object Recognition on a Reconfigurable Mesh

Concettina Guerra*
Dept. of Computer Sciences, Purdue University
West-Lafayette, 47907 IN, USA
and Dip. di Elettronica e Informatica, Università di Padova
via Gradenigo 6/a, 35131 Padova, Italy
email: guerra@era.dei.unipd.it, guerra@cs.purdue.edu

Abstract
This paper presents an approach to recognizing two-dimensional multiscale objects on a reconfigurable mesh architecture with horizontal and vertical broadcasting. The object models are described in terms of a convex/concave multiscale boundary decomposition that is represented by a tree structure. The problem of matching an observed object against a model is formulated as a tree matching problem. A parallel dynamic programming solution to this problem is presented that requires $O(\max(n,m))$ time on $n \times m$ reconfigurable mesh, where $n$ and $m$ are the sizes of the two trees.

1 Introduction

Model-based object recognition is an important part of a high level vision system. Given a model from a database of models, the problem is to find occurrences of the model in a given image. This is a difficult problem because of occlusion and possible overlap of parts. Furthermore, noise due to sensing inaccuracies, poor contrast, shadows, etc. complicate the task.

Many sequential approaches to matching in recognition systems have been considered in the literature [5]. Parallel implementations of some of the sequential algorithms have been proposed (see [6] for a survey).

In this paper we discuss a parallel implementation of a matching strategy based on multiscale tree descriptions of the objects. The domain we consider is that of 2D objects from digitized images. Objects are represented by their boundary contours at different levels of resolution. The contours in turn are represented by set of features, such as convex/concave segments. A tree structure is used to link features at different levels of resolution that are spatially related. The tree describes an object contour with increasing levels of details from the root down to the leaves.

The matching problem is formulated as a tree matching problem: find the best set of corresponding nodes at all levels of two trees subject to the constraint that only one node is mapped along each path from the root to a leaf. The time complexity of the sequential matching algorithm, based on dynamic programming, is $O(n \times m)$, where $n$ and $m$ are the number of model and image features, that is the number of nodes in the two trees [4]. In this paper, we show that the algorithm can be efficiently implemented in parallel on a $n \times m$ mesh with row and column broadcasting in time $O(\max(n,m))$.

Other multiscale models have been proposed in the literature [12], [14] that differ in terms of data structures used to organize the multiresolution image data, in the choice of the information associated with the elements of the structures and in the matching strategy. Our sequential matching algorithm improves on a similar approach to matching shapes [14] that is quadratic in the product of the number of segments at the finest resolution. Furthermore, unlike other multiscale approaches, the algorithm lends itself naturally to an SIMD parallel implementation.

The paper is organized as follows. In the next section we briefly review broadcasting in reconfigurable meshes. Section 3 describes the object multiscale tree model. Section 4 defines the problem statement and presents our solution. Section 5 describes the parallel algorithm.

2 Reconfigurable meshes

A reconfigurable mesh of size $n^2$ consists of an $n \times n$ array of processors where each processor has four switches connected to broadcast buses, as shown in Fig. 1. The switches are individually and locally controllable, allowing the mesh to be reconfigured into smaller sub-meshes according to the communication requirements of the algorithm. A processor is identified by its coordinates $(i, j)$ and its four ports are denoted by N (North), S (South), W (West), and E (East). A port can be connected to any other port. Thus there are 15 connection patterns. For instance, port N can be connected to S or N can be connected to both E and W. By disconnecting vertical (N and S) ports, broadcasting along

*Partially supported by the National Research Council of Italy and by the ESPRIT III Basic Research Programme of the EC under contract No. 9072 (Project GEPPCOM).
the $n$ separate rows can be obtained. Column broadcasting and sub-row and sub-column broadcast can be achieved with a proper setting of the switches. (Fig. 2). There exist a few variations of the basic model of reconfigurable mesh. A survey of the different models and of their power and flexibility is in [2]. Many algorithms for solving problems in image processing, computational geometry, have been designed for reconfigurable meshes [1], [2], [7], [8], [10], [11]. Some of these algorithms use a reconfigurable mesh with only broadcasts along row and column buses.

The algorithm presented in this paper uses the weaker version of a reconfigurable mesh in which only row and column broadcasts are possible. It is assumed that the delay along columns and rows is constant. The mesh operates in SIMD mode, that is all processors execute the same instruction at the same time; however, the selection of the switching configuration is done locally by each processor.

3 Multiscale Tree Representation

In [3] we introduced a method to derive a tree representation of a given planar object from multiresolution data. The method first obtains a description of the object at each level of resolution by decomposing its boundary into segments, having different lengths. A segment is convex or concave. The boundary decomposition is obtained by a simulation of the heat-diffusion process that is very suitable to parallel implementation. Two attributes are attached to each boundary feature to provide a measure of dissimilarity between mapped segments: the length and the degree of symmetry [3]. Boundary descriptions at different scales induce a tree description. Each node in the tree corresponds to a segment at some level of resolution; an arc connects segments at consecutive levels that are spatially related. More precisely, the children of a node correspond to consecutive segments at the next level of resolution that can be seen as one global segment at the level of the parent node. The left-to-right order of the children of a node corresponds to a given orientation of the curve boundary. Thus, the children of a node give a more detailed description of the same portion of the bound-
ary as the parent node. The leaves of the tree correspond to the finest representation of the shape. In Fig. 3, shapes of a human profile are given along with the corresponding tree representation. Full (empty) nodes of the tree correspond to convex (concave) segments.

The construction of the tree representation of an object from its boundary descriptions at all scales is obtained by a simple dynamic programming algorithm. For each two consecutive scales, the algorithm tries to find the correspondence between the extreme points of the segments of the two boundaries that minimizes the sum of Euclidean distances between corresponding points. Let \( P_1, P_2, \ldots, P_s \) and \( P'_1, P'_2, \ldots, P'_r \) be the sequences of points given by the extremes of the boundary segments at two consecutive scales in the clockwise direction. The node that corresponds to segment \( P_iP_{i+1} \) either has a single child \( P'_kP'_{k+1} \), or a number of children \( P'_kP'_{k+1} \ldots P'_{k+t} \). In the former case \( P_i \) and \( P_{i+1} \) are mapped into \( P'_k \) and \( P'_k \), respectively, in the latter into \( P'_k \) and \( P'_k \), respectively. The problem of finding a mapping that minimizes the sum of distances of corresponding points is similar to the problem of computing the edit distance between two sequences and can be solved by the dynamic programming technique in \( O(s^2 + r^2) \) time. The algorithm can be efficiently implemented on an SIMD architecture. A parallel implementation of the dynamic programming algorithm applied to the string editing problem exists that requires \( O(s + r) \) time on an SIMD mesh (or a linear array).

4 Tree matching

Let \( T \) and \( T' \) be two ordered trees associated to a model and an observed object. We want to determine the best set of corresponding nodes in the two trees. The mapping or matching must satisfy the following: 1) A node of \( T \) can be mapped to a node at any level of \( T' \); 2) the mapping has to preserve the left-to-right order of the nodes; 3) in each tree, the subtrees rooted at the mapped nodes are disjoint; 4) for any leaf, one of its ancestors (including the leaf itself) must be in the mapping.

In other words, 3) states that if a node is in the mapping, none of its ancestors is; this combined with 4) implies that there is exactly one node in the mapping along each path from a leaf node to the root of \( T \) (\( T' \)). It follows that the number of matched nodes is less or equal to the minimum number of leaves between \( T \) and \( T' \).

Intuitively, this means that for any segment of a model boundary at the finest level of resolution there must be one segment of the observed object at some resolution level covering it that is in the mapping.

The above definition of a mapping between trees can be modified to make the computation more robust in the presence of occlusion. When a portion of the shape boundary is not visible, there will be leaves that will not be covered by nodes having a corresponding one in the other tree. To allow
for unmatched portions of the boundary the constraint 4) can be replaced by the following: for any path from a leaf node to the root of each of the two trees there is at most one node in the mapping.

Different definitions of mappings between trees have been considered in different contexts [9], [13]. Our definition is tailored toward our specific application.

Let $n$ and $m$ be the number of nodes of $T$ and $T'$. Let $l$ the number of leaves and $leaves(T)$ the set of leaves of tree $T$. $T[i]$ denotes the node of $T$ whose position in the post-order traversal of $T$ is $i$. In the post-order, $T[1], T[2], ..., T[i]$ is in general a forest. $anc(i)$ is the set of ancestor nodes of $T[i]$, including $i$ itself. The postorder number of the father of node $T[i]$ is denoted by $p(i)$. $ll(i)$ denote the postorder number of the leftmost leaf, of the subtree rooted at $T[i]$; $ll(i) = i$ when $T[i]$ is a leaf node.

A matching between two trees $T$ and $T'$ is a set of integer pairs $M = \{(i_1, j_1), (i_2, j_2), ..., (i_k, j_k)\}$, $k \leq \text{Min}(l, l')$, $1 \leq i_1 \leq n$, $1 \leq j_1 \leq m$. Our goal is to find a matching which is optimal with respect to a given cost function and satisfies the above constraints:

- If $(i, h)$ is in $M$, then no $(i', k)$, with $i' \in \text{anc}(i)$, is in $M$.
- For any leaf $i$, there is a pair $(i', k)$, $i' \in \text{anc}(i)$, in $M$.

A measure of dissimilarity or distance $d(i, j)$ between nodes $i$ and $j$ is defined as follows. $d(i, j) = \infty$ if the labels (convex, concave) attached to the nodes are different; otherwise, $d(i, j)$ is the weighted sum of the differences of attributes of boundary segments corresponding to nodes $i$ and $j$. In the expression of $d(i, j)$ there is also a term that penalizes correspondences at coarser levels. This term is a function of the height of the two nodes and of the number of descendant nodes.

The problem can now be formulated as a minimization problem: find the set $M$ of pairs $(i, j)$ that minimizes the total distance function, that is

$$G(T, T') = \text{Min}_M \sum d(i, j)$$

Let $g(i, j)$ $1 \leq i \leq n$, $1 \leq j \leq m$ be the distance between the two forests $T[1], T[2], ..., T[i]$ and $T'[1], T'[2], ..., T'[j]$. It is $g(i, j) = G(T, T')$ when $i = n$ and $j = m$. Let $g(0, 0) = 0$ and $g(0, j) = g(i, 0) = \infty$ for $i \neq 0$, $j \neq 0$. Then

- if $i \neq p(i - 1)$ and $j \neq p(j - 1)$
  $$g(i, j) = \min\{g(i - 1, j), g(i, j - 1), g(i - 1, j - 1) + d(i, j)\};$$

Figure 4: The matching set $M = \{(i_1, j_1)\}$ of two trees

- if $i = p(i - 1)$ and $j \neq p(j - 1)$
  $$g(i, j) = \min\{g(i - 1, j), g(i, j - 1), g(i - 1, j - 1) + d(i, j)\};$$

- if $i \neq p(i - 1)$ and $j = p(j - 1)$
  $$g(i, j) = \min\{g(i - 1, j - 1), g(i - 1, j), g(i - 1, j - 1) + d(i, j)\};$$

The above recurrences specify the value of $g(i, j)$ in terms of three neighboring values $g(i - 1, j)$, $g(i, j - 1)$, $g(i - 1, j - 1)$ and of a fourth value $g(ll(i) - 1, ll(j) - 1)$. We call this latter pair $(ll(i) - 1, ll(j) - 1)$ the conjugate of $(i, j)$.

These relations suggest the use of dynamic programming to solve the minimization problem. When computing $g(i, j)$, $1 \leq i \leq n$, $1 \leq j \leq m$, all the values in the right side of the recurrence relation above have already been computed. Thus it takes constant time to compute $g(i, j)$ from the above leading to a $O(n \times m)$ time algorithm. The space complexity is also quadratic.
The edges directed into the grid point of the tree nodes. The graph has a vertex for every pair of tree problem. In the figure we draw the points such that point trees.

Thus, the number of out-edges of each node is three plus the same conjugate correspond to nodes along the same two

Figure 5 shows an example of a graph associated to the tree problem. In the figure we draw the points such that point trees.

Each grid point \( (i, j) \) of nodes of \( T \) and \( T^\prime \) and has at most four in-edges directed into the grid point \( (i, j) \): from the adjacent grid points \( (i - 1, j), (i, j - 1), (i - 1, j - 1) \), and from its conjugate \( (ll(i) - 1, ll(j) - 1) \). It easy to check that every pair \( (i, j) \) has at most one conjugate. Furthermore, all pairs with the same conjugate correspond to nodes along the same two paths in the two trees each starting from a leaf node and moving up towards the root following left branches only.

Thus, the number of out-edges of each node is three plus a number bounded by the product of the depths of the two trees.

Figure 5 shows an example of a graph associated to the tree problem. In the figure we draw the points such that point \( (i, j) \) is at the \( i \)th row from the top and \( j \)th column from the left. The edges directed into \( (i, j) \) from its three adjacent grid points \( (i - 1, j), (i, j - 1), \) and \( (i - 1, j - 1) \) are not drawn in the figure.

The following property is also easy to show. If \( (i, j) \) has conjugate \( (i', j') \), then, for any pair \( (k, l) \) with \( i' \leq k \leq i \) and \( j' \leq l \leq j \) we also have \( i' \leq k' \leq i \) and \( j' \leq l' \leq j \).

Each grid point \( (i, j) \) can be seen as lower right corner of a submatrix whose upper-left corner is \( (ll(i) - 1, ll(j) - 1) \). The above property states that the submatrices defined by two pairs and their corresponding conjugates are either nested or disjoint.

The matching problem can be seen as one of finding an optimal path in the associated graph from the vertex \((0, 0)\) to the vertex \((n, m)\). Such a problem can be solved by dynamic programming.

The parallel implementation we consider uses a mesh of processors with a processor associated to every vertex of the graph and labeled with the same integer pair as the graph node. The model of computation is that of a reconfigurable mesh with row and column broadcasting. The computation proceeds in a waveform fashion and consists of \( k = n + m \) stages. At each stage, a computation phase is followed by a broadcasting phase. During the computation phase of stage \( k \), the values \( g(i, j), i + j = k \), are derived from the above recurrences. The values \( g(i - 1, j) \) and \( g(i, j - 1) \) and \( g(i - 1, j - 1) \), needed in the computation of \( g(i, j) \), are easily accessible in constant time. As for the value \( g(ll(i) - 1, ll(j) - 1) \) corresponding to the conjugate of \((i, j)\), it was generally computed at earlier stages of the processing in a distant processor.

To make the values of the conjugate pairs available to all processors when needed, after each computation phase the algorithm uses the following broadcast phase. Consider a vertex \((p, q)\), with \( p + q = k \) and such that \( p + 1 \in leaves(T) \) and \( q + 1 \in leaves(T') \). We assume that each processor in the mesh has a flag indicating whether the tree nodes of the corresponding pair are both leaves. Furthermore, every processor contains the conjugate of its corresponding pair of nodes. This information can be pre-computed in sequential \( O(max(n, m)) \) time. After the value \( g(p, q) \), has been computed during the computation phase of stage \( k \), it is sent to all processors that will need it in later stages, i.e. to processors \((r, s)\), with \( ll(r) = p + 1 \) and \( ll(s) = q + 1 \).

More precisely, processor \((p, q)\) prepares a message containing the following: the pair \((p, q)\) from which the message originated and the value \( g(p, q) \). The broadcast is obtained by an alternating sequence of horizontal and vertical operations. First, the message is broadcast along row \( p \) to the \( E \) direction. Each processor \((p, s)\), with \( ll(s) = q + 1 \), stores the received message and becomes active. A similar operation is repeated along column \( q \), in the \( S \) direction, at the end of which all processors \((r, q)\), with \( ll(r) = p + 1 \), become active. Then, all active nodes of row \( p \) initiate a broadcast of the stored message to the \( S \) direction. Finally, all active nodes in column \( q \) do a broadcast operation to \( E \). At the end of these operations, the processors that have received two messages originated from the same processor are the ones that have \((p, q)\) as conjugate. All other processors delete the messages received.

Broadcast operations from different points along the same wave can be carried out in parallel. Consider at stage \( k \), broadcasts from \((p, q)\) and \((u, v)\), \( p + q = u + v = k \). Let us assume that \( p < u \). Then \( q > v \). Consider the first horizontal operation involving rows \( p \) and \( u \). Since the sets of nodes of \( T' \) with distinct leftmost leaves \( q + 1 \) and \( r + 1 \) in their corresponding subtrees are disjoint, the destination pairs of the two messages belong to different columns. Thus there is at most one active processor in each column as a result of the horizontal broadcasts. Simi-
larly, the active processors after the broadcast along different columns belong to different rows. As a consequence, at most two messages from active processors are received at the end by each processor. Each processor can determine in constant time whether the two messages originated from the same point (its conjugate). Thus the algorithm runs in time \( O(\max(n, m)) \).

Acknowledgments
The author wishes to thank A. Apostolico and M. Maresca for the helpful discussions.

References