A Transputer System for the Recognition of Human Faces
by Labeled Graph Matching

Rolf P. Würtz  Jan C. Vorbrüggen  Christoph von der Malsburg

Abstract

The method of labeled graph matching has been introduced as an extension to both associative memory and classical neural networks. The concept’s ability to tackle cognitive tasks is demonstrated here with a programme that recognizes images of human faces. Its parallel implementation on a network of transputers in OCCAM achieves recognition times below one minute.

1 General method

We are going to give only a very brief overview of the idea of labeled graph matching here. For an in-depth discussion, see [1]. Abstract objects are represented by labeled graphs. Once edge and vertex labels have been chosen in a way that they describe the essential features of an object while discarding unimportant ones, the problem of recognition is reduced to graph matching. Unlike with conventional neural networks, relations between objects can again be coded as labeled graphs, without any need for new cells. This is an answer to the serious problem of combinatorial explosion.

Our system for face recognition is meant to demonstrate the capabilities of this concept, which is in no way limited to vision. Currently, it should not be regarded as a mature mechanism for object recognition because it still lacks many ingredients, most prominently figure-ground separation and invariance to lighting conditions.

In contrast to the style of AI, our system needs no specialization to the application domain. It has been tested with equal success on the recognition of faces and of vehicles.

2 Image representation

A crucial part of the method is the right choice of the vertex labels. We transform the image \( I(\tilde{z}) \) by convolution with so-called Morlet wavelets [2]. Their definition and the related linear transform are as follows:

\[
\psi_{k} := n_{k,\sigma} \exp \left( -\frac{k^2 \tilde{z}^2}{2\sigma^2} \right) \exp \left( ik\tilde{z} \right)
\]

(1)

\[
(WI)(\tilde{k}, \tilde{z}_0) := \int \psi_{k}(\tilde{z}_0 - \tilde{z}) I(\tilde{z}) \, d^2 \tilde{z},
\]

(2)

where \( \tilde{k}, \tilde{z}, \tilde{z}_0 \in \mathbb{R}^2 \). This family of functions is well adapted to our problem for the following three reasons: 1) They are optimally localized in both space and frequency domain [3]. 2) The shape of the wavelets and the process of convolution are justified by neurophysiological results (see, e.g., [4]). Burr et al. build a vision model based on the squared sum of the responses pairs of simple cells with even and odd symmetry (absolute value of the complex convolution) and provide psychophysical evidence for the applicability of this model to human vision [5]. 3) The self similarity inherent in the definition of wavelets [2] makes the transform commute with translations, rotations and scaling of the image domain, although this is, of course, limited by the sampling density.

In order to improve matching behaviour, we take the modulus of \( WI \) and normalize within frequency levels (\( \phi \) is the direction angle of \( \tilde{k} \)):

\[
JI(\tilde{k}, \tilde{z}_0) = \frac{|WI|}{\int |WI| \, d^2 \tilde{z} \, d\phi}.
\]

(3)

The restriction of \( JI \) to some fixed \( \tilde{z}_0 \) is a local descriptor of the image near this point and will be referred to as a jet. These jets provide the vertex labels for our graphs, the vertices themselves being points in the image. The vertex set will be denoted by \( V \), the edge set by \( E \).

*This work was made possible by a grant from the German Ministry of Science and Technology (ITR-8800-H1). Contributions were made from grant 88-0274 by AFOSR.

1Max-Planck-Institut für Hirnforschung, Deutscherordensstr. 46, D-6000 Frankfurt 71, F.R.G.

2Dept. of Computer Science, University of Southern California, Los Angeles, CA 90089-0782, USA
Figure 1: Objects, images, and graphs. a) Sample object: Picture of a face, overlayed with the graph of stored jets. (For visualization, only nearest neighbour connections are shown; the actual graph is a complete one.) b) Picture of the same person, but looking in a different direction. The overlayed graph is the final result of the matching process with the object in a). c) Picture of a different person and the graph matched to a). In our system, this process is repeated for a number of stored objects, and the one giving the best value of the minimized cost function is selected. Final values of $C_{total}$ for a), b), c) are $-70.0$, $-57.7$ and $-55.8$, respectively. d) A jet from the tip of the nose in a) was selected and compared with all jets of the same picture. The result is displayed as a grey-level image, black coding for the best match. This gives an impression of the “cost surface” which drives the minimization process.
3 Object comparison

Similarity between jets is measured by means of a function $S(J^O_i J^I)$. Currently we are using one of the simplest choices, the cosine of the angle between two jets, as derived from the scalar product. Similarities are added up over all pairs of corresponding points in the two graphs. The total cost function contains in addition a topology term:

$$C_{total} := \lambda C_{top} + C_{jet} = \lambda \sum_{(i,j) \in E} (\tilde{A}_{(i,j)} - \tilde{A}_O^{i,j})^2 - \sum_{i \in V} S(J^O_i J^I).$$  \hspace{1cm} (4)

$\tilde{A}_{(i,j)}$ is the vector connecting vertices $i$ and $j$ in the image domain, $\tilde{A}_O^{i,j}$ in the object domain. The factor $\lambda$ controls the relative weight of the topology constraint. $C_{total}$ is minimized by using the Metropolis algorithm at temperature zero. We terminated minimization once a certain number of consecutive moves failed to improve it. For examples of resulting graphs, see figure 1.

4 General Purpose Farm Software

We implemented the programme on a network of transputers, using the OCCAM 2 language. Grey-level pictures are acquired with a transputer-controlled frame grabber. The network consisted of up to 16 T800s.

The three main components — two-dimensional FFT, Morlet transform, and matching the object database to an image — all consist of a number of independent tasks which operate on a subset of the data. This makes them well suited to parallelization using the farm concept, where a controller hands out tasks to any number of identical, anonymous workers.

The typical chain of events when using a farm is as follows:

- The controller asks the user what to do next.
- It broadcasts any global (read-only) data required to all workers. This will include information on what action was requested.
- A “run” is started, which consists of a set of tasks. Each worker is handed a task; when it has returned the results, it is given a new one, until there are no more tasks available. At this point, the controller waits for any remaining results to come in.

This sequence is repeated until the programme is terminated. Note that one user-requested action may require a number of runs by the farm.

To make life easier for the developer, we designed a set of routines which implement a general purpose farm. The user just provides two modules (one for the controller, one for the workers) which use a simple and safe interface via OCCAM channels. On the controller, the farm software takes care of distributing tasks to the workers and collecting the results, routing messages as necessary. It also gives the user control of acquisition and display of performance data (see section 6). On the worker processors, the farm software provides an interface to the routing system and handles the read-out of monitoring data. It also allows the reception and buffering of the next command from the router while the previous one is being worked on; thus, an optimal overlap of computation and communication is achieved.

5 Network Topology and Routing Software

Suited to the structure of a farm is some sort of tree topology, with the controller at its root. In order to achieve maximal flexibility, the router system supports any tree-like structure, the only limitation being the number of transputer links. A network reconfiguration can be accomplished without changes to the user software, only the network description has to be modified.

After initialization is complete, each router runs two parallel processes. The up process receives messages coming in from all active “down” links and from the local worker process. It then sends these messages on its “up” link. The up process on the controller hands all incoming messages to the farm software, which passes them on to the user. The down process receives message from the “up” link (or from the farm software, in case of the controller processor). It examines the address field of every message and then passes it on to either the local worker or, after examining its routing table, to the appropriate “down” link. If the address has the reserved value 0, it is a broadcast packet and is sent down all active links and to the local worker.
Table 1: Performance of parallel FFT and Morlet transform implementation for different numbers \( N \) of worker processors. A sampling interval of 3 ms was used; this has an overhead of less than one percent. FFT: Times given are for both forward and inverse FFT of an image of size 128 × 128 pixels containing a vertical stripe pattern. The serial part is calculated as 0.08 s. Note the impact of increased controller utilization on the relative speedup, especially for 12 processors. It limits the useful network size to about 16 processors. Morlet transform: Times given include the time to perform the forward FFT of the data. Measurement conditions are as above. The serial part is calculated as 1.37 s. In this case, the bottleneck is the communication of the results from the workers to the controller.

6 Performance Figures
The Morlet transform (2) was based on a set of kernels with 6 values of \( |\vec{k}| \), spaced by half octaves, each frequency level containing 8 orientations differing by 22.5°, for a total of 48 convolutions. They were computed using the relation \( \psi_{k} * I = \mathcal{F}^{-1}(\mathcal{F}\psi_{k} \cdot \mathcal{F}I) \). Table 1 gives some performance indicators for our programme. The two-dimensional FFT was parallelized as two runs, one performing all one-dimensional row FFTs, followed by all column FFTs. In the Morlet transform, it is used to compute \( \mathcal{F}I \), and every worker calculates \( \mathcal{F}\psi_{k} \) from the analytical formula. Every task then consists of multiplication with the appropriate kernel, the inverse FFT, and normalization (see (3)) of the result. Elapsed time was measured using the transputer clock. To determine load balancing, a statistical programme counter sampling technique was used. The load on the controller, measured in this way, is reported in table 1.

Timing for the graph matching phase depends strongly on the objects and their number. As a reference, a single Monte Carlo step takes ~ 2.5 ms. One match requires between 2500 and 5000 steps, of which some 10% improve \( C_{\text{total}} \). Matching a database of 30 objects on 12 transputers thus takes about 20 seconds.

Acknowledgments: We would like to thank Wolf Singer for providing excellent working conditions in Frankfurt. Peter König, Jörg Lange, and Joachim Buhmann contributed to this work through innumerable hints and discussions.

References