DIMACS Technical Report 2002-45
October 2002

Serial and tree-serial dynamic programming with application to face identification

by

Vadim Mottl
Department of Cybernetics
Tula State University
Lenin Ave. 92, 300600 Tula, Russia
mottl@uic.tula.ru

Ilya Muchnik
DIMACS
Rutgers University
P.O. Box 8018, Piscataway, NJ 08855
muchnik@dimacs.rutgers.edu

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, NEC Research Institute and Telcordia Technologies (formerly Bellcore).

DIMACS was founded as an NSF Science and Technology Center, and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

This report continues the series of DIMACS Technical Reports on the variational approach to constructing algorithms of data analysis [1,2,3] that consists in systematically exploiting the fact that all the kinds of decision making rest on the intent, at least, a mental one, to minimize an appropriate measure of discrepancy between the observed data set and result of its processing in the form of a model of the original data set to be evaluated. The specificity of the optimization problems adequate to a majority of practical data analysis problems is that the variables constituting the sought-for data set model are associated with the nodes of a graph that determines its a priori structure. As a rule, the objective function to be minimized turns out to be pair-wise separable in accordance with this graph, i.e. be the sum of partial functions of one and two variables associated with nodes and edges of graph.

However, for a separable objective function of general kind, the problem of global optimization does not lend itself to an effective algorithmical solution until the variable adjacency graph is assumed to be a tree. For this latter kind of objective functions, the original optimization problem breaks up into a hierarchy of partial problems each of which consists in the optimization of a set of functions of only one variable. The proposed principle of solving tree-like pair-wise separable optimization problems on the basis of a hierarchical elimination of variables is called in this Report the tree-serial dynamic programming as a generalization of the classical serial dynamic programming.

Such a structure of the interaction of variables in an objective function is an intermediate special case between easily solvable serial problems and challenging nonserial ones. It is shown, that such a moderate generalization, on the one hand, retains practically all the computational advantages of the serial problems and, on the other, easily allows for using it, with a bit of heuristics, in solving typical image processing problems.

As a practical problem, we address in this work the problem of face recognition from a single image. The tree-serial dynamic programming is applied to fulfill a nonlinear transformation of one image plane relative to another by spatially constrained elastic matching of two pixel grids for the purpose of featureless face identification by immediate measuring image similarity. The method provides the linear computational complexity with respect to the number of pixels without application of parallel computers.

It will be shown in the next Report that the proposed approach is adequate, along with the problem of face identification, also the problems of face detection and tracking in an image flow and, so, covers the entire concept of dynamic vision.

Contents

1 Introduction 4

2 Serial dynamic programming 10

2.1 The succession of Bellman functions and the procedure “forward then back” 10
2.2 Marginal functions 11
2.3 Left-side and right-side Bellman functions: A modified serial dynamic programming procedure “forward and counter-forward” 11
2.4 Computational complexity of the serial dynamic programming in the case of discrete variables 12

3 Tree-serial dynamic programming 13

3.1 Structure of a tree-serial objective function 13
3.2 Bellman functions and the tree-serial dynamic programming procedure 15
3.3 Computational complexity of the tree-serial dynamic programming procedure 16

4 Elastic transformation of the image pixel grid for similarity-based face identification 17

4.1 Featureless approach to face identification 17
4.2 Elastic transformation of the image pixel grid versus elastic graph matching 17
4.3 The basic pair-wise separable criterion of elastic transformation 19
4.4 Partial tree-serial criteria for single columns 19
4.5 Algorithm of image matching: minimization of partial tree-serial objective functions 20

5 Support object classifiers with elastic kernel functions for face identification 22

5.1 Structure of support-object decision rules of face identification 22
5.2 Rigid and elastic kernel functions 23
5.3 Data base and experimental results 25

6 Conclusions 26

7 Acknowledgements 27

References 27

1 Introduction

It was even Leonard Euler who marked that everything what happens in the world bears the sense of some minimum or maximum. Concerning data analysis, the verity of this observation, which procured the name of the variational principle, manifests itself in the fact that all the kinds of decision making rest on the intent, at least, a mental one, to minimize an appropriate measure of discrepancy $J(X \mid Y) \to \min_{X \in \mathsf{X}}$ between the observed data set $Y \in \mathsf{Y}$ and result of its processing within the permissible range $X \in \mathsf{X}$. The variational principle turns a practical problem into the formal one of minimizing the respective objective function and shifts off the burden of heuristically constructing the data analysis algorithm into the tilled field of the optimization theory.

We will consider a data set Y as an experimentally acquired function y_t, $t \in T$, which takes values from an appropriate range $y \in \mathsf{Y}$, where index (argument) $t \in T$ runs through all the elements of the data set.

What is typical for a large class of applications is the fact that the elementary information units y_t are ordered along the axes of one t or several arguments $t = (t_1, ..., t_k)$, so that the data collection subject to an analysis as a whole is rather an array than set. Each element of such an array is naturally associated with a number of its neighbors which may be defined in several ways depending on the nature of the particular data source, but in any case a neighborhood relation on the set of data elements $t \in T$ is an inherent feature of the data array. It is convenient to express the neighborhood relation in the form of an undirected graph G which is understood as the set of neighboring element pairs $G \subset T \times T$.

The evident neighborhood graph for signals is chain (Fig. 1,a), the simplest neighborhood relation usually considered as appropriate for an image pixel grid is two-dimensional rectangular lattice (Fig. 1,b), an image flow suggests its being viewed as three-dimensional lattice (Fig. 1,c). A less obvious kind of neighborhood between data set elements is tree (Fig. 1,d), but, as it will be shown below, the tree-like neighborhood is a convenient heuristics that allows for essential simplification of data-processing algorithms.

In this work, we address the sufficiently wide class of data analysis problems in which it is required, for all the elements of the data set $t \in T$, to estimate the parameters $x_t \in \mathsf{X}_t$ of some local data models constituting jointly the entire model of the data set $X = (x_t, t \in T)$ that presupposes some a prioiri relationship between the values of single parameters. Data analysis problems of such a kind can be represented as problems of transforming the original array $Y = (y_t, t \in T)$, $y_t \in \mathsf{Y}$, into another function $X = (x_t, t \in T)$, which would be defined on the same set of the elements of the array $t \in T$ and take values $x_t \in \mathsf{X}_t$ from a set specific, in the general case, for each element t.

For instance, a typical problem of image analysis is photometric normalization, when it is required to substitute the given visual image $Y = (y_{\mathbf{t}}, \mathbf{t} \in T)$, $T = \{\mathbf{t} = (t_1, t_2), t_1 = 1,...,N_1, t_2 = 1,...,N_2\}$ by another one $X = (x_{\mathbf{t}}, \mathbf{t} \in T)$ which would retain the main features of the original but suppress effects of specific lighting conditions (Fig. 2). Another problem of image analysis is that of measuring the similarity of two given images $Y' = (y'_{\mathbf{t}}, \mathbf{t} \in T)$ and $Y'' = (y''_{\mathbf{t}}, \mathbf{t} \in T)$ by finding such a displacement vector field between the two pixel grids $X = (\mathbf{x}_{\mathbf{t}}, \mathbf{t} \in T)$, $\mathbf{x}_{\mathbf{t}} = (x_{\mathbf{t}}^{(1)}, x_{\mathbf{t}}^{(2)})$ that would point the identical elements to be immediately

compared $Y' = (y'_{\mathbf{t}}, \mathbf{t} \in T)$ and $Y''(X) = (y''_{\mathbf{t}+\mathbf{x}_{\mathbf{t}}}, \mathbf{t} \in T)$ (Fig. 3). A sequence of video frames $Y = (Y_1, ..., Y_N)$, $Y_\tau = \left(y_{\mathbf{t},\tau},\ \mathbf{t} = (t_1, t_2)\right)$ taken from a changing or moving object is an example of a three-dimensional data array, in which it is required, for instance, to track the apparent motion produced by projecting the original object on the shooting plane. It is natural to express the result of tracking as a three-dimensional array of two-dimensional displacement vectors $X = (X_1, ..., X_{N-1})$, $X_\tau = \left(\mathbf{x}_{\mathbf{t},\tau},\ \mathbf{t} = (t_1, t_2)\right)$, so that each element of the current frame $Y_\tau = (y_{\mathbf{t},\tau}, \mathbf{t} \in T)$ would be associated with its match in the next $Y_{\tau+1}(X_\tau) = (y_{\mathbf{t}+\mathbf{x}_{\mathbf{t},\tau},\tau+1}, \mathbf{t} \in T)$.

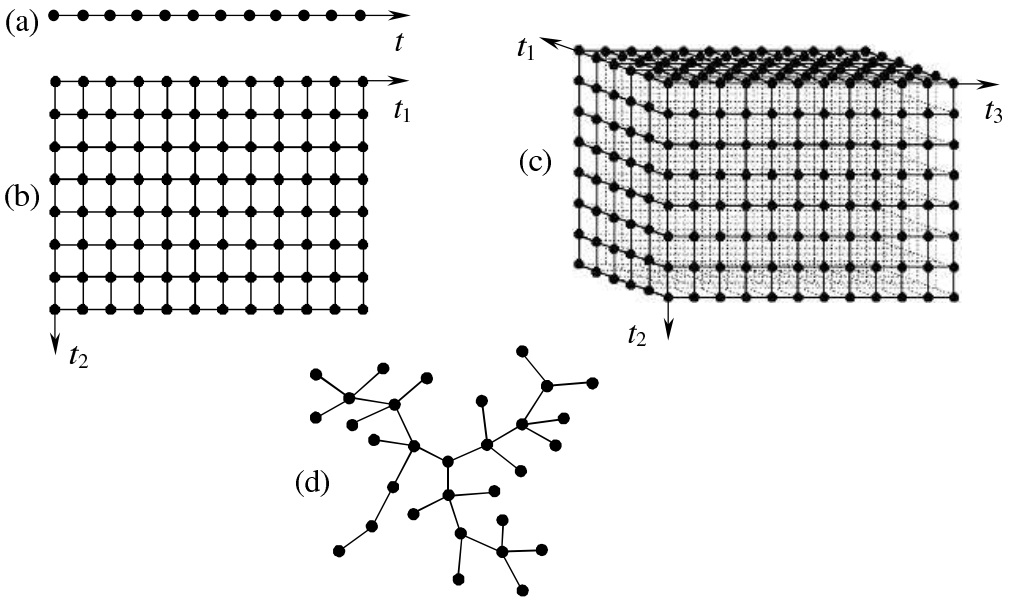


Figure 1. Neighborhood graphs on the set of data set elements: chain (a), two-dimensional lattice (b), three-dimensional lattice (c), and tree (d).

Figure 2. The original face image Y (left) and the result of photometric normalization X (right).

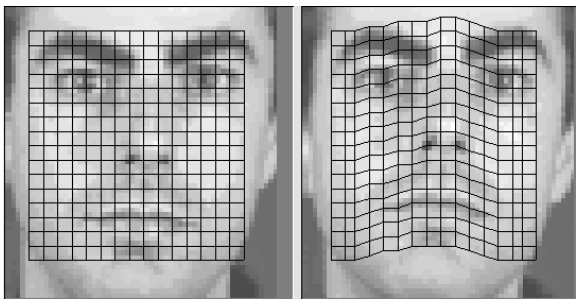


Figure 3. Two shots of the same face Y' and Y'' taken under different camera angles and the displacement field X of matching the left shot to the right one.

In accordance with the variational approach, the sought-for secondary data array $X=(x_t, t\in T)$ must occur as argument in an objective function $J(X\,|\,Y)$ that should assess the discrepancy between each admissible version of the result X and the given data array Y. The range of definition of the objective function $X\in \mathsf{X}$ is completely determined once the ranges of the elementary variables $x_t\in \mathsf{X}_t$ are chosen. Thus, the result of data analysis $\hat{X}(Y)$ is defined by the condition

$$\hat{X}(Y)=\arg\min_{x_t\in \mathsf{X}_t} J(X\,|\,Y). \qquad (1)$$

The specificity of the optimization problems under consideration is that the variables constituting the data set model to be evaluated $X=(x_t, t\in T)$ are associated with the nodes of the neighborhood graph G on the set of data array elements T. It is typical for the majority of practical problems that the objective function $J(X)=J(X\,|\,Y)$ is pair-wise separable in accordance with this graph, i.e. is sum of partial functions of one $\psi_t(x_t)$, $t\in T$, and two variables $\gamma_{t't''}(x_{t'},x_{t''})$, $(t',t'')\in G$, associated with nodes and edges of graph G and called, respectively, node and edge functions [1,4]:

$$J(X)=\sum_{t\in T}\psi_t(x_t)+\sum_{(t',t'')\in G}\gamma_{t't''}(x_{t'},x_{t''})\to \min_{x_t\in \mathsf{X}_t,\, t\in T}. \qquad (2)$$

As a rule, the node functions are data-dependent $\psi_t(x_t)=\psi_t(x_t\,|\,Y)$ and penalize the discrepancy between the assumed values of the local parameters at the respective positions and the data set, whereas the model-dependent edge functions $\gamma_{t't''}(x_{t'},x_{t''})$ express the pair-wise a priori preferences in accordance with the accepted graph-based parametric data model by penalizing the tentative parameter values in pairs. However, in some cases, the edge functions may be data-dependent as well $\gamma_{t't''}(x_{t'},x_{t''})=\gamma_{t't''}(x_{t'},x_{t''}\,|\,Y)$, this fact is out of significance from the computational point of view.

The computational complexity of the optimization problem (2) substantially depends on the kind of the adjacency graph G. In the simplest case, when the variable adjacency graph is a chain (Fig. 1,a),

$$J(X)=\sum_{t=1}^{N}\psi_t(x_t)+\sum_{t=2}^{N}\gamma_t(x_{t-1},x_t), \qquad (3)$$

where $N=|T|$ is the number of variables, there exists a highly effective means of optimization, namely, the classical dynamic programming procedure [5,6] that provides the linear computational complexity with respect to N as result of decomposition of the original objective function of the full number of variables into a succession of intervening function each of only one variable, that are to be minimized sequentially.

Since the variables of a chain-like separable objective function (3) are arranged into a series, Bellman's classical principle of building optimization algorithms that consists in sequential solving the problem by eliminating the variables one by one is called in [7] the serial dynamic programming.

But the serial dynamic programming is inapplicable, at least, in a straightforward way, to optimization problems with more complex graphs of variable adjacency like rectangular lattice (Fig. 1,b,c).

The pair-wise separable objective function (2) whose variables are adjacent in accordance with a binary graph G, including the special case (3), is, in its turn, a special case of the more general class of objective functions $J(X)=J(x_t, t\in T)$ studied in [7]. Let $\mathsf{G}=\{T'\subset T\}$ be a

system of subsets of the variable set T, so that $\bigcup_{T' \in \mathsf{G}} T' = T$, and $\gamma_{T'}(X') = \gamma_{T'}(x_t, t \in T')$ be partial objective functions each of which depends only on the respective subset of variables. It is assumed that the full objective function is the sum of partial functions

$$J(X) = \sum_{T' \in \mathsf{G}} \gamma_{T'}(X'). \tag{4}$$

If the set of variables T is considered as a set of nodes, then the set of its subsets $\mathsf{G} = \{T' \subset T\}$ is a hypergraph on T, namely, graph in which generalized edges T' called hyperedges may connect more than two nodes [8,9]. We will refer to an objective function having the structure (4) as separable objective function of general kind, and the hypergraph G as its variable adjacency hypergraph. In the special case when each hyperedge of a hypergraph G consists only of two nodes $T' = (t', t'')$, we obtain a usual binary graph G and, respectively, a pair-wise separable objective function (2).

Unfortunately, the search for an optimization algorithm for separable objective functions with variable adjacency hypergraph G of general kind (4) leads to computational difficulties, and pair-wise separability (2) makes no exemption. The nature has put a fundamental barrier to the hopes to construct a non-iterative procedure which would provide finding the global minimum or maximum of a separable function with an arbitrary structure of variable adjacency, and would be less computationally dissipative than exhaustive search. The variety of known algorithmical means of finding the global optimum of a separable function of general kind is limited to the procedures of iterative random search like stochastic relaxation and simulated annealing [10,11]. Iterative procedures of such a kind converge especially slowly, because the large number of random steps is just that computational expedient which prevents the search process from sticking in local extrema. The low computational speed is restricting for the range of applications where such algorithms might be used.

Fortunately, the hypergraph G of the variables' occurrence in a separable objective function, or, in case the objective function is pair-wise separable, graph G of their pair-wise adjacency, are specific in each particular practical problem, what allows, as a rule, for a computational simplification of the optimization algorithm. The methodology of exploiting the specificity of group-wise variable adjacency in each particular optimization problem on the basis of Bellman's optimality principle is called in [7] the nonserial dynamic programming. The main idea of nonserial dynamic programming is application of Bellman's principle to any optimization problem $J(X) = J(x_t, t \in T) \to \min_{x_t \in \mathsf{X}_t}$ by sequential elimination of variables in that or other order. Three modes of elimination are considered in [7], namely, (a) eliminating the variables one by one in accordance with a preset ordering $T = \{1,2,...,N\}$, (b) forming an ordered succession of blocks of variables $T = T_1 \cup T_2 \cup ... \cup T_K$, $T_i \cap T_j = \varnothing$, for sequential elimination of variables in blocks, and (c) multilevel elimination that comprises block-wise elimination recursively applied to the partial optimization problems inside the blocks. The problem of finding the most advantageous ordering, grouping or hierarchical grouping of the variables of the given separable objective function with the purpose to provide the minimal amount of enumeration, or the minimal volume of memory required for storing the backward recurrent relations, or both of that, is called in [7] the secondary optimization problem. Finding the optimal mode of variable elimination is an extremely difficult problem even for the simplest one-by-one technique.

In this work, we focus on the special case of a pair-wise separable objective function (2). Finding the strict solution of the secondary optimization problem remains to be a challenge if the variables are assumed to be adjacent in accordance with an arbitrary graph G. But in a still

more special case, when this graph has no cycles, i.e. is a tree, the order of hierarchical elimination of the variables becomes evident, and the secondary optimization problem does not exist at all. For this latter kind of objective functions, the original optimization problem with $|T|$ variables breaks up into a hierarchy of $|T|$ partial problems each of which consists in the optimization of a set of functions of only one variable.

By its structure, the hierarchical elimination of variables inherits the main computational steps of the serial dynamic programming by performing them in a hierarchical order, therefore, we call the proposed principle of solving tree-like pair-wise separable optimization problems the tree-serial dynamic programming.

In the case of a pair-wise separable objective function of general kind, the advantages of the tree-like organization of the variable adjacency suggests a solution to the secondary optimization problem for the block-wise nonserial elimination technique by grouping the variables, i.e. nodes of the variable-adjacency graph G, in such a way that the resulting aggregated graph would be a tree with as small size of the maximal group as possible.

The main aim of this work is to show that the tree-serial dynamic programming offers a powerful computational means for efficiently solving a wide class of image processing problems that includes, in particular, the problems of smoothing and image matching. These problems have been traditionally considered from the statistical viewpoint as those of estimating the hidden Markov component of a two-component random field in which the original image plays the role of the observable component [12,13]. The assumed Markov property of the hidden field $X = (x_{\mathbf{t}}, \mathbf{t} \in T)$ was meant to express the a priori smoothness constraints of that or other kind, which usually follow from the nature of the respective practical problem. However, the essence of both advantages and difficulties of such an approach lies, actually, not in its probabilistic aspect, but in the fact that the procedures of estimating the realization of the hidden random field are those of solving a specific optimization problem $\hat{X}(Y) = \arg\min(or\ \max) J(X \mid Y)$.

The specificity of this problem is that the variables constituting the vector to be evaluated $X = (x_{\mathbf{t}}, \mathbf{t} \in T)$ are associated with the elements of the image pixel grid $T = \{\mathbf{t} = (t_1, t_2)\}$, and that, due to the principal Markov assumption, the objective function $J(X \mid Y)$ is separable, at least, in the general sense (4). The specific composition of hyperedges constituting the hypergraph G depends on how the a priori properties of the hidden Markov field are defined, namely, first, the vicinity of each element $\mathbf{t} \in T$, and, second, the conditional Markov distribution of the hidden value $x_{\mathbf{t}}$ with respect to the values of other elements in this vicinity. In particular, if the vicinity of an element $\mathbf{t}$ is understood as the set of elements immediately adjacent with it in accordance with the pixel-neighborhood graph G, usually, rectangular lattice (Fig. 1,b), and the conditional probability distribution of $x_{\mathbf{t}}$ is defined as product of efficients each dependent only on one of the neighboring values, the objective function $J(X \mid Y)$ will be pair-wise separable (2) with the variable adjacency graph G.

As noted above, the latter fact itself does not contribute essentially to an improvement of the computational solvability of the optimization problem, until the variable adjacency graph is assumed to be a tree. The idea of approximating a random field by a tree-like Markov one by way of building the minimum spanning tree, proposed in [14], is inapplicable to the image pixel grid, because it is clear that there is no way of replacing the lattice-like graph by a single tree without loss in the ability to ensure preservation of pair-wise a priori preferences in all directions from each point of the image plane. An alternative idea of tree approximation to a Markov random field proposed in [15] consists in using an individual tree for evaluating each

value of the hidden field. As a compromise between these two extremes, we use here, just as in [1,4], an individual pixel neighborhood tree, which is defined, nevertheless, on the whole pixel set and shares the same horizontal branches with other trees (Fig. 4). The respective generalized image processing procedure is aimed at finding the optimal values only of the goal variables at the stem nodes in each tree.

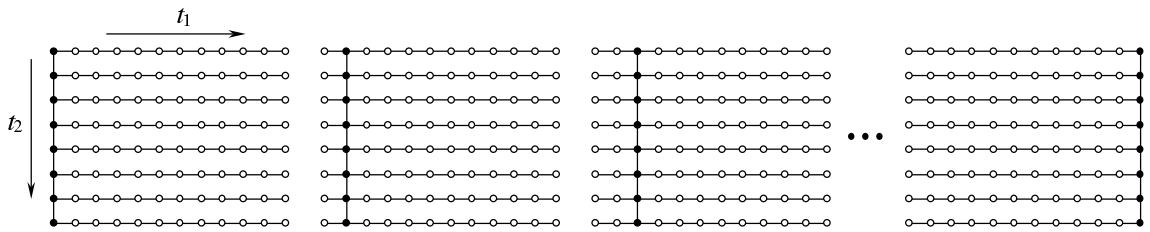


Figure 4. Adjacency graphs of partial objective functions with tree-like separability for evaluating the goal variables in each single vertical row of the image pixel grid.

The title of this work points at images and image flows as the particular kind of data sets primarily addressed as objects of processing. However, the mathematical framework considered here and the approach to constructing data processing algorithms following from it are applicable to a much more wide range of data sets.

More specifically, the problem indicated in the title as the goal of data processing is that of face recognition from a single image or an image flow [16]. We keep here to the similarity-based pattern recognition technique [17,18] which is based on the notion of the so-called kernel function $K(Y',Y'')$ meant to measure similarity between any two feasible objects, in our case, images. Normally, face images Y are represented on the pixel grid $T=\{\mathbf{t}=(t_1,t_2),\ t_1=1,...,N_1,\ t_2=1,...,N_2\}$ as gray-level vectors $Y=(y_{\mathbf{t}},\mathbf{t}\in T)$. The simplest version of a kernel function is the inner product of the respective gray-level vectors on the pixel grid $K(Y',Y'')=\sum_{t\in T} y'_t y''$. However, inevitable distortions caused by differences in camera viewing angles and facial expression will worsen the error rate of recognition, because identical coordinates y'_t and y''_t of two gray-level vectors Y' and Y'' will correspond to incomparable points of the faces.

A widely adopted way of overcoming this obstacle is elastic graph matching, that consists in establishing correspondence between a few number of selected fiducial points forming, in each of two face images, a graph with nodes represented by appropriate local features of brightness distribution around them. The criterion of elastic graph matching is pair-wise separable (2) with lattice-like adjacency of variables, and, to overcome inevitable computational difficulties of its optimization, a special neural network architecture is built to implement the iterative principle of simulated annealing [19]. In this work, the computational advantages of the tree-serial dynamic programming applied to a series of partial pixel-neighborhood trees instead of the full lattice allows for immediate matching all the elements of the image pixel grids in contrast to a few nodes of the image model graphs. In this case, it is more appropriate to speak about an elastic transformation of the image pixel grid as a whole rather than about elastic graph matching [20].

2 Serial dynamic programming

2.1 The succession of Bellman functions and the procedure "forward then back"

In this Section, we discuss the optimization problems that arise when the variational approach is used to synthesize algorithms for the analysis of signals $Y=(y_t, t_{\min}\le t\le t_{\max})$, whose essential feature consists in the fact that they are ordered along the axis of some argument, as a rule, the time or space coordinate. In particular, a single row of an image can be treated as a signal. The results of our consideration will be applicable to a wide class of practical problems in which it is required, for all points of the axis of a discrete argument $t=1,...,N$, to estimate the current value of sufficiently smoothly changing parameter x_t of a certain local model. The sequence of these estimates $X=(x_t, t=1,...,N)$ is just the desired result of signal processing.

In this case, the variational principle of model estimation leads to a pair-wise separable objective function with serial adjacency of variables (**3**). The natural procedure for the optimization of objective functions of such a kind is the classical method of dynamic programming [5,6], which consists in replacing the initial problem of finding the minimum of a function of many arguments by a succession of substantially less difficult problems of minimizing intermediate functions of one argument. Since the arguments occur in the initial pair-wise separable objective function (**3**) in series, and the intermediate optimization problems are to be naturally solved in the same series, the classical method of dynamic programming is called in [7] the serial dynamic programming.

The central idea of the serial dynamic programming is the concept of Bellman functions

$$\tilde{J}_t^-(x_t)=\min_{x_s\in X_s,\, s<t} J_t^-(x_s, s\le t), \tag{5}$$

which are related to partial objective functions

$$J_1^-(x_1)=\psi_1(x_1),\ J_t^-(x_s, s\le t)=\sum_{s=1}^{t}\psi_s(x_s)+\sum_{s=2}^{t}\gamma_s(x_{s-1},x_s),\ t=2,...,N, \tag{6}$$

where the upper index "minus" points at the fact that the functions are defined from left to right along the axis of argument numbers. Since $J_t^-(x_s, s\le t)=J_{t-1}^-(x_s, s\le t-1)+\psi_t(x_t)+\gamma_t(x_{t-1},x_t)$ for $t=2,...,N$, the Bellman functions can be sequentially computed by the rule

$$\tilde{J}_t^-(x_t)=\psi_t(x_t)+\min_{x_{t-1}\in X_{t-1}}\left[\gamma_t(x_{t-1},x_t)+\tilde{J}_{t-1}^-(x_{t-1})\right],\ t=2,...,N, \tag{7}$$

called the forward recurrent relation, starting from $\tilde{J}_1^-(x_1)=\psi_1(x_1)$. The inverse form of the forward recurrent relation

$$\tilde{x}_{t-1}(x_t)=\arg\min_{x_{t-1}\in X_{t-1}}\left[\gamma_t(x_{t-1},x_t)+\tilde{J}_{t-1}^-(x_{t-1})\right] \tag{8}$$

will be referred to as backward recurrent relation.

Obviously, $\hat{x}_N=\arg\min_{x_N}\tilde{J}_N^-(x_N)$ is the optimal value of the last variable, and the other optimal values are easily found by the rule $\hat{x}_{t-1}=\tilde{x}_{t-1}(\hat{x}_t)$, $t=N,...,2$, if the backward recurrent relations (8) were stored when the Bellman functions (7) were recurrently recalculated in the forward order.

This procedure is the classical serial dynamic programming procedure, which we will refer to as the "forward then back" procedure.

2.2 Marginal functions

We introduce here an additional denotation $\hat{J}_t(x_t)$ for the so-called marginal functions [1] each of which shows how the full objective function depends on the value of a single variable if the other variables take conditionally optimal values with respect to x_t:

$$\hat{J}_t(x_t) = \min_{x_s \in X_s,\, s \neq t} J(X). \tag{9}$$

For the last variable, the marginal function is found at the last step of the forward pass $\hat{J}_N(x_N) = \tilde{J}_N^-(x_N)$. The following theorem shows how all the remaining marginal functions can be found.

Theorem 1. *Let the marginal function be known for a variable x_t, $t > 1$. Then, for the immediately preceding variable, the marginal node function satisfies the relation*

$$\hat{J}_{t-1}(x_{t-1}) = \min_{x_t \in X_t} \left\{ \hat{J}_t(x_t) + \left[\gamma_t(x_{t-1}, x_t) + \tilde{J}_{t-1}^-(x_{t-1}) \right] - \left[\gamma_t(\tilde{x}_{t-1}(x_t), x_t) + \tilde{J}_{t-1}^-(\tilde{x}_{t-1}(x_t)) \right] \right\}. \tag{10}$$

This theorem is an evident special case of the more general Theorem **4** which will be formulated below in Section 3.2.

It is clear that the optimal value of each variable is the minimum point of the respective marginal function $\hat{x}_t = \arg\min_{x_t \in X_t} \hat{J}_t(x_t)$, and, so, the marginal functions carry information on the "local" behavior of the objective function at its minimum point when only one of the arguments is varied. Theorem 1 gives a backward recurrent relation that, being applied along with (8), enables deriving all the marginal functions in addition to the optimal values $\hat{x}_t$, if required.

2.3 Left-side and right-side Bellman functions: A modified serial dynamic programming procedure "forward and counter-forward"

In some applications, the classical "forward then back" scheme of dynamic programming is insufficient [21]. The insufficiency of this scheme reveals itself when it is required to find the optimal values of a single variable $\hat{x}_t$ or a pair of immediately adjacent variables $\hat{x}_{t-1}$ and $\hat{x}_t$ several times for different versions of, respectively, the node function $\psi_t(x_t)$ or the edge function $\gamma_t(x_{t-1}, x_t)$. In this Section, we consider another scheme where two kinds of Bellman functions are used, instead of one, which are computed toward each other from opposite endpoints of the axis of variable numbers $t = 1,..., N$ and $t = N,..., 1$.

The partial criteria $J_t^-(x_s, s \le t)$ (6) are defined from left to right starting from the first (left) variable and will be called, hereafter, the left partial criteria. Along with them, it is quite natural to consider the right partial criteria

$$J_t^+(x_s, s \ge t) = \sum_{s=t}^{N} \psi_s(x_s) + \sum_{s=t}^{N-1} \gamma_{s+1}(x_s, x_{s+1}),\ t = 1,..., N-1,\quad J_N^+(x_N) = \psi_N(x_N), \tag{11}$$

which are symmetric to the left ones and defined from the end of the variable order.

Respectively, in addition to the left Bellman functions (5), we introduce their right analogs

$$\tilde{J}_t^+(x_t) = \min_{x_s \in X_s,\, s>t} J_t^+(x_s, s \geq t), \tag{12}$$

which are recurrently recalculated in the opposite direction

$$\tilde{J}_t^+(x_t) = \psi_t(x_t) + \min_{x_{t+1} \in X_{t+1}} \left[\gamma_{t+1}(x_t, x_{t+1}) + \tilde{J}_{t+1}^+(x_{t+1})\right],\ t = N-1,...,1, \tag{13}$$

starting from the last right Bellman function $\tilde{J}_N^+(x_N) = \psi_N(x_N)$.

The "meeting" of the left and the right Bellman functions at some point t directly yields the marginal function of the variable x_t (9). To formulate the respective theorem, it is convenient to extend the definition of edge functions $\gamma_t(x_{t-1}, x_t)$ and partial criteria (6) and (11) for one extra position to the left $t = 0$ and to the right $t = N+1$ outside the variable index series $t = 1,..., N$:

$$\gamma_1(x_0, x_1) \equiv 0,\ J_0^-(x_0) \equiv 0,\ J_{N+1}^+(x_{N+1}) \equiv 0. \tag{14}$$

Theorem 2. *For any pair-wise separable objective function* $J(X) = J(x_t, t = 1,..., N)$ *with serial adjacency of variables, the marginal function of the variable* x_t *can be represented, with respect to (14), as the sum*

$$\hat{J}_t(x_t) = \min_{x_{t-1} \in X_{t-1}} \left[\tilde{J}_{t-1}^-(x_{t-1}) + \gamma_t(x_{t-1}, x_t)\right] + \psi_t(x_t) + \min_{x_{t+1} \in X_{t+1}} \left[\gamma_{t+1}(x_t, x_{t+1}) + \tilde{J}_{t+1}^+(x_{t+1})\right]. \tag{15}$$

Proof. It immediately follows from (6) and (11) that for any t

$$J(x_t, t = 1,..., N) = J_{t-1}^-(x_s, s \leq t-1) + \gamma_t(x_{t-1}, x_t) + \psi_t(x_t) + \gamma_{t+1}(x_t, x_{t+1}) + J_{t+1}^+(x_s, s \geq t+1).$$

Then, in accordance with (9),

$$\hat{J}_t(x_t) = \min_{x_s \in X_s,\, s \leq t-1} \left[J_{t-1}^-(x_s, s \leq t-1) + \gamma_t(x_{t-1}, x_t)\right] + \psi_t(x_t) + \min_{x_s \in X_s,\, s \geq t+1} \left[\gamma_{t+1}(x_t, x_{t+1}) + \tilde{J}_{t+1}^+(x_{t+1})\right] =$$

$$\min_{x_{t-1} \in X_{t-1}} \left[\min_{x_s \in X_s,\, s<t-1} J_{t-1}^-(x_s, s \leq t-1) + \gamma_t(x_{t-1}, x_t)\right] + \psi_t(x_t) +$$

$$\min_{x_{t+1} \in X_{t+1}} \left[\gamma_{t+1}(x_t, x_{t+1}) + \min_{x_s \in X_s,\, s>t+1} \tilde{J}_{t+1}^+(x_{t+1})\right],$$

whence the equality (15) immediately follows on the force of (5) and (12). **Q.E.D.**

It remains only to find the optimal value of the variable x_t by the evident rule

$$\hat{x}_t = \arg\min_{x_t \in X_t} \hat{J}_t(x_t). \tag{16}$$

Thus, we have obtained an alternative scheme of the serial dynamic programming, which will be called the "forward and counter-forward" procedure. Two independent recurrent procedures (7) and (13) run in the opposite directions from the different ends of the variable series and yield jointly the marginal functions (15) and optimal values (16) of all the variables.

2.4 Computational complexity of the serial dynamic programming in the case of discrete variables

Since all the computational operations in the serial dynamic programming are performed sequentially in accordance with the order in which of the variables occur in the objective function, it is clear that the number of computational operations required for finding the optimal values of all the variables adds up from the groups of them related to single variables. If all the

variables take values from the same finite set $\mathsf{X}_t = \mathsf{X}$, the computational complexity of the serial dynamic programming will be linear with respect to the number of variables N.

The forward run of the "forward then back" procedure implies recurrent recalculation of Bellman functions (7) for the variables $t = 2,..., N$. Finding one of $|\mathsf{X}|$ values of a Bellman function $\tilde{J}_t^-(x_t)$, $x_t \in \mathsf{X}$, represented by a vector with $|\mathsf{X}|$ elements requires enumeration of $|\mathsf{X}|$ values of the preceding variable x_{t-1}, what makes all in all enumeration of $|\mathsf{X}|^2$ pair-wise combinations. When the last Bellman function $\tilde{J}_N^-(x_N)$ is found, the optimal value of the last variable $\hat{x}_N$ will be determined as result of enumeration of $|\mathsf{X}|$ tentative values $x_N \in \mathsf{X}$. Each of the backward recurrent relations (8) to be stored on the forward run is a vector with $|\mathsf{X}|$ integer elements, which immediately give the optimal values of all the variables, so that no enumeration is required on the backward run. Thus, the "forward then back" procedure requires enumeration of $(N-1)|\mathsf{X}|^2 + |\mathsf{X}|$ values of the goal variables.

The "forward and counter-forward" procedure consists of two independent runs of recurrent recalculation of, respectively, the left $\tilde{J}_t^-(x_t)$ and the right Bellman functions $\tilde{J}_t^+(x_t)$, whereupon the optimal value of each of N marginal functions $\hat{J}_t(x_t)$ is to be found. All in all, $2(N-1)|\mathsf{X}|^2 + N|\mathsf{X}|$ values of the goal variables are to be enumerated.

3 Tree-serial dynamic programming

3.1 Structure of a tree-serial objective function

In this section, we consider a generalization of the serial dynamic programming under the assumption that the graph of variable adjacency in a pair-wise separable objective function is more complex than chain, namely, has the form of a tree. Such a structure of the interaction of variables in an objective function is an intermediate special case between easily solvable serial problems and challenging nonserial ones. We shall see, that such a moderate generalization, on the one hand, retains practically all the computational advantages of the serial problems and, on the other, easily allows for using it, with a bit of heuristics, in solving typical image processing problems.

Let G be a tree on the set of nodes $T = \{t\}$ (Fig. 5), and

$$J(X) = \sum_{t \in T} \psi_t(x_t) + \sum_{(t',t'') \in G} \gamma_{t't''}(x_{t'}, x_{t''}), \quad x_t \in \mathsf{X}_t, \tag{17}$$

be a pair-wise separable objective function to be minimized. Functions of one variable $\psi_t(x_t)$ the nodes of the graph will be called node functions, and functions of two variables $\gamma_{t't''}(x_{t'}, x_{t''})$ assigned to the edges will be, respectively, called edge functions.

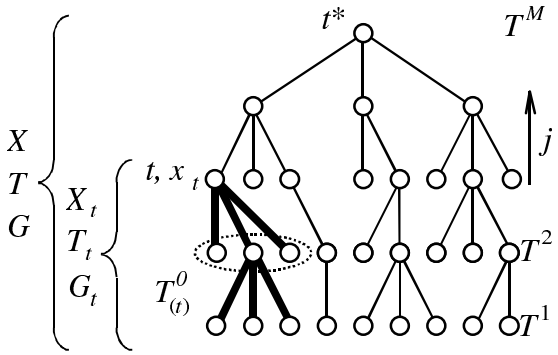


Figure 5. Structure of an arbitrary tree.

It is out of significance which of the nodes t^* is assigned the root of the tree, but once it is done, the set of nodes gets partitioned into a number of subsets each consisting of nodes of a certain level

$$T=\bigcup_{j=1}^{M}T^j\ ,\ T^j\bigcap T^k=\varnothing\ ,\ \ j\neq k\ .$$

We will denote the set of nodes of the lowest level by T^1, and the set containing the only node of the highest level, namely, the root, by T^M.

Let, further, the tree $G_t\subset G$ formed by a node t and its descendants be called descendant tree of this node. It is clear that $G_t=\varnothing$ for leaves, in particular, for all $t\in T^1$, and $G_t=G$ for the root $t\in T^M$. The set of all the nodes in the descendant tree of a node t will be denoted by T_t, symbol $T_{(t)}$ will mean the same set without this node itself, i.e. the set of its descendants, and $T^0_{(t)}\subseteq T_{(t)}$ will denote the subset of its immediate (next) descendants. So, $T_t=\{t\}$ and $T^0_{(t)}=T_{(t)}=\varnothing$ for leaves, and $T_{t^*}=T$ for the root.

Analogously, symbols $X_t=(x_s,\,s\in T_t)$ and $X_{(t)}=(x_s,\,s\in T_{(t)})$ will mean partial vectors of variables at the respective sets of nodes.

Finally, let us associate with each node t a partial objective function

$$J_t(X_t)=\sum_{s\in T_t}\psi_s(x_s)+\sum_{(s',s'')\in G_t}\gamma_{s's''}(x_{s'},x_{s''})$$

which has absolutely the same structure as the full objective function (17) but is defined on the respective partial descendant tree. It is easy to see that

$$J_t(X_t)=J_t(x_t,X_{(t)})=\psi_t(x_t)+\sum_{s\in T^0_{(t)}}\gamma_{ts}(x_t,x_s)+\sum_{s\in T^0_{(t)}}J_s(X_s)=$$

$$\psi_t(x_t)+\sum_{s\in T^0_{(t)}}\{\gamma_{ts}(x_t,x_s)+J_s(x_s,X_{(s)})\}.$$

In particular, for the nodes of the lowest level $t\in T^1$, as well as for all other leaves, the partial objective function gets trivial $J_t(X_t)=J_t(x_t)=\psi_t(x_t)$, and for the root the partial function is nothing else than the full objective function $J_{t^*}(X_{t^*})=J(X)$ (17).

3.2 Bellman functions and the tree-serial dynamic programming procedure

The principal idea of the optimization procedure for separable objective functions supported by trees is based, like the classical dynamic programming procedure for chains, on the notion of Bellman function

$$\tilde{J}_t(x_t) = \min_{X_{(t)}} J(x_t, X_{(t)}) = \psi_t(x_t) + \min_{X_s,\, s\in T^0_{(t)}} \sum_{s\in T^0_{(t)}} \left\{\gamma_{ts}(x_t, x_s) + \tilde{J}_s(x_s)\right\}, \quad x_t \in \mathsf{X}_t. \tag{18}$$

Theorem 3. *For all the nodes of the tree, the Bellman functions satisfy the recurrent relation*

$$\tilde{J}_t(x_t) = \psi_t(x_t) + \sum_{s\in T^0_{(t)}} \min_{x_s\in \mathsf{X}_s} \left\{\gamma_{ts}(x_t, x_s) + \tilde{J}_s(x_s)\right\}. \tag{19}$$

Proof. It immediately follows from the definition of Bellman function (18) that $\tilde{J}_t(x_t) = \psi_t(x_t) + \min_{X_s,\, s\in T^0_{(t)}} \sum_{s\in T^0_{(t)}} \{\gamma_{ts}(x_t, x_s) + J_s(X_s)\}$.

Each of the vector variables $X_s = (x_s, X_{(s)})$, $s \in T^0_{(t)}$, occurs only in one summand $\{\gamma_{ts}(x_t, x_s) + J_s(X_s)\} = \{\gamma_{ts}(x_t, x_s) + J_s(x_s, X_{(s)})\}$, therefore, the operations of minimization with respect to $(x_s, X_{(s)})$ can be fulfilled independently of each other before summation:

$$\tilde{J}_t(x_t) = \psi_t(x_t) + \sum_{s\in T^0_{(t)}} \min_{X_s} \{\gamma_{ts}(x_t, x_s) + J_s(X_s)\} =$$

$$\psi_t(x_t) + \sum_{s\in T^0_{(t)}} \min_{x_s\in \mathsf{X}_s} \left\{\gamma_{ts}(x_t, x_s) + \min_{X_{(s)}} J_s(X_{(s)})\right\}.$$

Here, in their turn, $\min_{X_{(s)}} J_s(X_{(s)})$ are Bellman functions $\tilde{J}_s(x_s)$ of the node variables x_s, whence (19) immediately follows. **Q.E.D.**

The fundamental property of the Bellman function (19) established by this theorem will be called the upward recurrent relation. Once the Bellman functions are known for all the immediate descendants $s \in T^0_{(t)}$ of a node t, the relation defines the Bellman function at this node. The inverted form of the upward recurrent relation

$$\tilde{x}_s(x_t) = \arg\min_{x_s\in \mathsf{X}_s} \left\{\gamma_{ts}(x_t, x_s) + \tilde{J}_s(x)\right\}, \quad s \in T^0_{(t)}, \tag{20}$$

will be referred to as the downward recurrent relation.

The optimization procedure is based on the assumption that there exists a sufficiently effective way of solving the partial optimization problems occurring in (19), and storing, in a compact form, the Bellman functions or downward relations.

The procedure runs twice through all the nodes of the graph in order of their membership in the hierarchy levels T^j, first upwards $j = 1, \ldots, M$, and then downwards $j = M, \ldots, 2$.

It starts with the nodes of the lowest (first) hierarchy level $t \in T^1$, $j = 1$. As $T^0_{(t)} = \varnothing$ for these nodes,

$$\tilde{J}_t(x_t) = \psi_t(x_t), \quad x_t \in \mathsf{X}_t, \quad t \in T^1. \tag{21}$$

Thereupon, the upward recurrent relation (19) is applied sequentially to the nodes of the higher levels $j = 2,..., M$. The Bellman functions $\tilde{J}_t(x_t)$ or downward relations $\tilde{x}_s(x_t)$, $x_t \in \mathsf{X}_t$, are to be stored for all the nodes of the tree except those of the first level.

The Bellman function at the root $\tilde{J}_{t^*}(x_{t^*})$, $x_{t^*} \in \mathsf{X}_{t^*}$, that is obtained at the last step of the upward pass, immediately gives the optimal value of the root variable:

$$\hat{x}_{t^*} = \arg\min_{x_{t^*} \in \mathsf{X}_{t^*}} \tilde{J}_{t^*}(x_{t^*}). \tag{22}$$

On the downward pass, as the procedure descends the hierarchy levels $j = M,..., 2$, the already found optimal values of the current level determine optimal values of the variables at the immediately underlying level in accordance with the downward recurrent rules (20) stored as the result of the upward pass or determined by the stored Bellman functions (19)

$$\hat{x}_s = \tilde{x}_s(\hat{x}_t), \quad s \in T^0_{(t)}. \tag{23}$$

The degenerate particular case of a tree is tree without branching nodes, i.e. a chain. In this case, the above described optimization procedure becomes the classical serial dynamic programming procedure considered above in Section 2.

We introduce here the notion of the marginal node functions $\hat{J}_t(x_t)$ by full analogy with the marginal functions in the serial dynamic programming (9):

$$\hat{J}_t(x_t) = \min_{x_s \in \mathsf{X}_s,\ s \in T,\ s \neq t} J(X). \tag{24}$$

At the root, the marginal node function is found at the last step of the upward pass $\hat{J}_{t^*}(x_{t^*}) = \tilde{J}_{t^*}(x_{t^*})$. The following theorem gives a downward recurrent relation that allows deriving all the remaining marginal node functions from the obvious one at the root.

Theorem 4. *Let the marginal node function be known for an inner (not-leaf) node t of the tree. Then, for each of its immediately descendant nodes the marginal node function satisfies the relation*

$$\begin{aligned} \hat{J}_s(x_s) = \min_{x_t \in \mathsf{X}_t} \Big\{ \hat{J}_t(x_t) + \big[\tilde{J}_s(x_s) + \gamma_{ts}(x_t, x_s)\big] - \min_{x'_s \in \mathsf{X}_s} \big[\tilde{J}_s(x'_s) + \gamma_{ts}(x_t, x'_s)\big] \Big\} = \\ \min_{x_t \in \mathsf{X}_t} \big\{ \hat{J}_t(x_t) + \big[\tilde{J}_s(x_s) + \gamma_{ts}(x_t, x_s)\big] - \big[\tilde{J}_s(\tilde{x}_s(x_t)) + \gamma_{ts}(x_t, \tilde{x}_s(x_t))\big] \big\}. \end{aligned} \tag{25}$$

The proof of the theorem is to be found in [1].

3.3 Computational complexity of the tree-serial dynamic programming procedure

Despite the nonserial organization of variable adjacency in a tree-serial objective function, the computational operations in the tree-serial dynamic programming are performed sequentially, just as in the serial dynamic programming.

To find the minimum of a tree-serial objective function $J(x_t, t \in T)$, we have, for each edge of the tree $(t,s) \in G$, to calculate the value of the function $\{\gamma_{ts}(x_t, x_s) + \tilde{J}_s(x_s)\}$ (19) for each combination of values of the adjacent variables $x_t \in X_t$ and $x_s \in X_s$, i.e. $|\mathsf{X}|^2$ times if all the variables take values from the same finite set $\mathsf{X}_t = \mathsf{X}$. All in all, we will have to enumerate $K|\mathsf{X}|^2$ pair-wise combinations of variable values, where K is the number of edges in the tree G. When the Bellman function at the root $\tilde{J}_{t^*}(x_{t^*})$ is found, the optimal value of the last

variable $\hat{x}_{t^*}$ will be determined as result of enumeration of $|X|$ values $x_{t^*} \in X$. All in all, enumeration of $K|X|^2 + |X|$ values of the goal variables will be required. So, the computational complexity of the tree-serial dynamic programming remains the same as that of serial dynamic programming $(N-1)|X|^2 + |X|$, because $N-1$ is just the number of edges in the chain-like variable adjacency graph.

4 Elastic transformation of the image pixel grid for similarity-based face identification

4.1 Featureless approach to face identification

The facial appearance is one of a number of specific characteristics of a person that allows for visual identification of one individual against the others [22]. The general problem of face identification (authentication) is formulated as a problem of testing the null hypothesis that a given face belongs to one of a finite set of persons called clients against the alternative hypothesis that this is one of all other people conventionally called impostors. Within the bounds of the general problem, two modes are distinguished, namely face verification mode, when the alleged client claims his identity, and face recognition mode, in which the claimant does not introduce himself.

We keep here to the featureless pattern recognition technique [17] which, being applied to the problem of face identification, removes the necessity of extracting from a face image its discriminatory properties, which provide establishing the identity of the respective persons. Instead, it requires finding a pair-wise numerical relation between images, that would allow for building a decision rule for face identification by way of comparing the given face image with a number of basic images. The central notion of the similarity-based approach to pattern recognition is that of the so-called kernel function $K(Y', Y'')$ [18] meant to measure similarity between any two feasible objects, in our case, images.

Normally, face images Y are represented on the pixel grid $T = \{\mathbf{t} = (t_1, t_2), t_1 = 1,..., N_1, t_2 = 1,..., N_2\}$ as gray-level vectors $Y = (y_{\mathbf{t}}, \mathbf{t} \in T)$. The simplest version of a kernel function is the inner product of the respective gray-level vectors on the pixel grid $K(Y', Y'') = \sum_{t \in T} y'_t y''$. However, inevitable distortions caused by differences in camera viewing angles and facial expression will worsen the error rate of recognition, because identical coordinates y'_t and y''_t of two gray-level vectors Y' and Y'' will correspond to incomparable points of the faces.

4.2 Elastic transformation of the image pixel grid versus elastic graph matching

A widely adopted way of overcoming this obstacle is elastic graph matching One of the images Y' called the reference image is represented by the set of appropriate local features $(\mathbf{z}_{\mathbf{t}}(Y'), \mathbf{t} \in T')$ of brightness distribution around the nodes of a sparse subgrid $T' \subset T$ that covers the most fiducial points like eyes, nose, lips, etc., or, as the simplest choice, is evenly distributed over the pixel grid. The subgrid T' is considered as planer graph G' whose edges $(\mathbf{t}, \mathbf{s}) \in G'$ correspond to neighboring nodes in accordance with a binary neighborhood relation.

In the other image Y'' called the probe one, an analogous graph G'' is formed by displaced elements of the pixel grid $T'' = \{\mathbf{t} + \mathbf{x}_\mathbf{t}, \mathbf{t} \in T'\}$ as result of elastic deformation of the reference graph, and at nodes of G'' the same local features are measured $\left(\mathbf{z}_{\mathbf{t}+\mathbf{x}_\mathbf{t}}(Y''), \mathbf{t} \in T'\right)$.

The essence of elastic graph matching is finding the deformation $\hat{X} = (\hat{\mathbf{x}}_\mathbf{t}, \mathbf{t} \in T')$ of the graph in the probe image G'' relative to G' by minimizing a combined criterion

$$J(X \mid Y', Y'') = \sum_{\mathbf{t} \in T'} \| \mathbf{z}_\mathbf{t}(Y') - \mathbf{z}_{\mathbf{t}+\mathbf{x}_\mathbf{t}}(Y'') \|^2 + \beta \sum_{(\mathbf{t},\mathbf{s}) \in G'} \| \mathbf{x}_\mathbf{t} - \mathbf{x}_\mathbf{s} \|^2 \to \min. \qquad (26)$$

Here the first sum is the measure of residual local image dissimilarity around the matched nodes, the second sum penalizes the grid deformation, and coefficient β controls the desired rigidity of elastic matching. Then, the overall similarity between the images is expressed by the kernel function calculated as the sum of the inner products of the matched local feature vectors

$$K(Y', Y'') = \sum_{\mathbf{t} \in T'} \mathbf{z}_\mathbf{t}^T(Y') \mathbf{z}_{\mathbf{t}+\mathbf{x}_\mathbf{t}}(Y''). \qquad (27)$$

Finding the minimum point of this criterion is a hard computational problem which is solved by an iterative technique based on neural networks with dynamic link architecture [19]. The process of optimization is based on the principle of simulated annealing and requires a large amount of computing power, therefore, it is usually implemented on parallel computers. In the latter case, the time of calculations is proportional to $\log |T'|$ where $|T'|$ is the number of nodes in graph G'. Otherwise, when a usual computer is applied, the computational complexity will grow to $|T'| \log |T'|$.

It should be marked, that for the local features $\mathbf{z}_\mathbf{t}(Y')$ and $\mathbf{z}_{\mathbf{t}+\mathbf{x}_\mathbf{t}}(Y'')$ around, respectively, fiducial points in the reference image $\mathbf{t} \in T' \subset T$ and corresponding displaced points $\mathbf{t} + \mathbf{x}_\mathbf{t}$ in the probe one would be informative, the size of the window in which the features are evaluated must be large enough. But for these two values would be comparable, the form of each of the windows in the probe image should differ from that in the reference one in accordance with the final deformation of the probe-image plane $\hat{X} = (\hat{\mathbf{x}}_\mathbf{t}, \mathbf{t} \in T')$, which is unknown before the optimization of the criterion (26) is completed. Therefore, to avoid incomparability of local features, the window should be small in size, ideally, consist of only one pixel, what is senseless from the viewpoint of the amount of information carried by each single feature vector.

The greater the window, the longer time is required for calculating the local features. Besides, finding fiducial points of the reference image is also quite a time-consuming operation called segmentation of the reference image.

All these issues can be resolved by taking an everywhere dense fragment of the reference-image pixel grid as the set $T' \subset T$ of nodes of graph G' with single normalized brightness values at each of them $y'_\mathbf{t}$ as well as at displaced nodes $y''_{\mathbf{t}+\mathbf{x}_\mathbf{t}}$ of graph G'' in the probe image instead of feature vectors $\mathbf{z}_\mathbf{t}(Y')$ and $\mathbf{z}_{\mathbf{t}+\mathbf{x}_\mathbf{t}}(Y'')$. In this case, it is appropriate to speak about an elastic transformation of the image pixel grid as a whole rather than about graph matching.

However, the number of variables $X = (\mathbf{x}_\mathbf{t}, \mathbf{t} \in T')$ of the optimization problem (26) will grow dramatically, to such an extent that the usual iterative technique of elastic graph matching becomes unacceptable. It is just the aim of this work to propose and test a computationally economic procedure of elastic transformation of the image pixel grid on the basis of replacing the pixel-neighborhood lattice by a system of trees with application of the tree-serial dynamic programming to each of the resulting partial tree-like criteria.

4.3 The basic pair-wise separable criterion of elastic transformation

Let $T' \subset T$ be the inner part of the pixel grid of the reference image without a border of a preset width Δ along the edges. A mapping $T' \to T$ of the interior of the reference image into the whole pixel grid of the probe one can be expressed in the form $\mathbf{t} \in T' \to \mathbf{u}(\mathbf{t}) = \mathbf{t} + \mathbf{x}_\mathbf{t} \in T$, where $\mathbf{x}_\mathbf{t} = (x_\mathbf{t}^{(1)}, x_\mathbf{t}^{(2)}) \in \mathsf{F}$ are vectors with integer components

$$\mathsf{F} = \{(x^{(1)}, x^{(2)}) : -\Delta \le x^{(1,2)} \le \Delta\}. \tag{28}$$

Thus, the desired transformation of the pixel grid of the probe image is completely specified by the displacement vector field $X = (\mathbf{x}_\mathbf{t}, \mathbf{t} \in T')$.

On the one hand, the sought vector field X should provide as good agreement between the two images as possible $(y'_\mathbf{t} - y''_{\mathbf{t}+\mathbf{x}_\mathbf{t}})^2 \to \min$, where $y'_\mathbf{t}$ and $y''_{\mathbf{t}+\mathbf{x}_\mathbf{t}}$ are normalized brightness values (Fig. 2 in Introduction).

On the other hand, the transformation $T' \to T$ is to be chosen so that there would be no folds in the image plane of the probe image. In this work, we do not restrict the class of transformations. To prevent folds, we pose an "elastic" penalty upon the neighboring displacement vectors, so that the transformation as a whole $X = (\mathbf{x}_\mathbf{t}, \mathbf{t} \in T')$ would be smooth enough, namely, $\| \mathbf{x}_\mathbf{t} - \mathbf{x}_\mathbf{s} \|^2 \to \min$ for all $(\mathbf{t}, \mathbf{s}) \in G$, where G is the lattice-like graph of pixel adjacency within the interior of the reference image pixel grid (Fig. 1,b).

Finding a compromise between these two mutually contradictory requirements is achieved by optimizing a combined objective function

$$J(X \mid Y', Y'') = \sum_{\mathbf{t} \in T'} (y'_\mathbf{t} - y''_{\mathbf{t}+\mathbf{x}_\mathbf{t}})^2 + \beta \sum_{(\mathbf{t},\mathbf{s}) \in G} \| \mathbf{x}_\mathbf{t} - \mathbf{x}_\mathbf{s} \|^2 \to \min . \tag{29}$$

The number of its variables is equal to the number $|T'|$ of nodes in G, i.e. the number of pixels in the interior T' of the reference image. By its structure, this criterion ifs a pair-wise separable objective function with lattice-like adjacency of discrete variables (2). More specifically, the problem (29) is a problem of pair-wise separable integer quadratic optimization. We consider the desired elastic transformation as the displacement-vector field which provides the minimum value of the basic criterion (29):

$$\hat{X} = \begin{pmatrix} \mathbf{x}_{11} & \cdots & \mathbf{x}_{N_1 1} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{1 N_2} & \cdots & \mathbf{x}_{N_1 N_2} \end{pmatrix} = \arg\min J(X \mid Y', Y'') . \tag{30}$$

This is a nonserial optimization problem with pair-wise adjacency of variables in the form of a lattice, for which there exists no effective way of sequential elimination of variables, neither one by one nor in blocks.

4.4 Partial tree-serial criteria for single columns

To utilize the advantages of the tree-serial dynamic programming (Section 3), we minimize, instead of the full lattice-like objective function (29), (30), a series of partial tree-serial criteria $J^{(Tree, t_1)}(X \mid Y', Y'')$, $t_1 = 1, ..., N_1$, each of the same full set of variables $X = (\mathbf{x}_\mathbf{t}, \mathbf{t} \in T')$ but obtained from (29) by removing edge functions corresponding to vertical edges of the lattice except one column (Fig. 4):

$$J^{(Tree,\,t_1)}(X \mid Y', Y'') = \sum_{t_1=1}^{N_1}\sum_{t_2=1}^{N_2}(y'_{\mathbf{t}} - y''_{\mathbf{t}+\mathbf{x}_{\mathbf{t}}})^2 + \beta_1\sum_{t_1=2}^{N_1}\sum_{t_2=1}^{N_2}\| \mathbf{x}_{t_1-1,\,t_2} - \mathbf{x}_{t_1t_2} \|^2 + \beta_2\sum_{t_2=2}^{N_2}\| \mathbf{x}_{t_1,\,t_2-1} - \mathbf{x}_{t_1t_2} \|^2 \to \min, \tag{31}$$

where upper index $(Tree, t_1)$ is meant to indicate that the adjacency of variables in this criterion is determined by the tree with t_1 th column as the stem.

Let $\hat{X}^{(Tree,\,t_1)}\left(\hat{\mathbf{x}}^{(Tree,\,t_1)}, t_1 = 1,..., N_1, t_2 = 1,..., N_2\right)$ be the vector field obtained as the minimum point of criterion (31) for the t_1 th stem

$$\hat{X}^{(Tree,\,t_1)} = \arg\min J^{(Tree,\,t_1)}(X \mid Y', Y''),$$

and $\left(\mathbf{x}_{t_11}^{(Tree,\,t_1)},..., \mathbf{x}_{t_1N_2}^{(Tree,\,t_1)}\right)$ be the t_1 th column cut out of it. The main heuristic idea of reducing calculations when forming the displacement-vector field of elastic transformation is to assemble the displacement vector field from these single partially optimal stem-columns:

$$\hat{X} = \begin{pmatrix} \mathbf{x}_{11}^{(Tree,\,1)} & \cdots & \mathbf{x}_{11}^{(Tree,\,N_1)} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{1N_2}^{(Tree,\,1)} & \cdots & \mathbf{x}_{N_1N_2}^{(Tree,\,N_1)} \end{pmatrix}. \tag{32}$$

4.5 Algorithm of image matching: minimization of partial tree-serial objective functions

It remains to specify how to solve the series of optimization problems (31). Let us consider a single row t_2 of the tree $(\mathbf{x}_{1t_2},..., \mathbf{x}_{N_1t_2})$ (Fig. 4) and associate it with a still more partial objective functions than (31), namely, a pair-wise separable function with chain-like separability of variables

$$J^{(Row,\,t_2)}(\mathbf{x}_{1t_2},..., \mathbf{x}_{N_1t_2}) = \sum_{t_1=1}^{N_1}(y'_{t_1t_2} - y''_{t_1+x_{t_1t_2}^{(1)},\,t_2+x_{t_1t_2}^{(2)}})^2 + \beta_1\sum_{t_1=2}^{N_1}\| \mathbf{x}_{t_1-1,\,t_2} - \mathbf{x}_{t_1t_2} \|^2 \to \min, \tag{33}$$

where upper index (Row, t_2) is meant to indicate that the arguments of this function form the t_2 th row in the pixel-neighborhood lattice.

If we give the t_1 th variable of function (33) a fixed value $\mathbf{x}_{t_1t_2}$ and minimize it by other variables, the resulting minimum will be a function of only one variable:

$$\hat{J}_{t_1}^{(Row,\,t_2)}(\mathbf{x}_{t_1t_2}) = \min_{\mathbf{x}_{s_1t_2}:\,s_1 \neq t_1} J^{(Row,\,t_2)}(\mathbf{x}_{1t_2},..., \mathbf{x}_{N_1t_2}). \tag{34}$$

Such a function is the marginal function of the t_1 th variable with respect to the serial objective function (33) (Section 2.2). If the displacement vectors with integer components are varied within the bounds $\mathbf{x}_{t_1t_2} \in \mathsf{F}$ (28), each marginal function can be represented by a table with $(2\Delta+1)^2$ elements.

Below, we consider how to find all marginal functions $\hat{J}_{t_1}^{(Row,\,t_2)}(\mathbf{x}_{t_1t_2})$, $t_1 = 1,..., N_1$, $t_2 = 1,..., N_2$. So far, let us assume that this is done. Then, as it is shown in [1], all the sought displacement vectors in the t_1 th column of (32) can be found as the solution of the optimization problem

$$J^{(Col,\,t_1)}(\mathbf{x}_{t_11},..., \mathbf{x}_{t_1N_2}) = \sum_{t_2=1}^{N_2} \hat{J}_{t_1t_2}^{(Row)}(\mathbf{x}_{t_1t_2}) + \beta_2 \sum_{t_2=2}^{N_2} \| \mathbf{x}_{t_1,\,t_2-1} - \mathbf{x}_{t_1t_2} \|^2 \to \min. \qquad (35)$$

Here upper index (Col, t_1) means that the arguments are the elements of the t_1 th column of the pixel grid. Each of these chain-like pair-wise separable objective functions of integer variables $\mathbf{x}_{t_1t_2} \in \mathsf{F}$ can be minimized by the usual dynamic programming procedure.

Now about finding marginal functions $\hat{J}_{t_1}^{(Row,\,t_2)}(\mathbf{x}_{t_1t_2})$ (34). Let us define, in addition to the serial objective function (33), also two truncated versions of it, namely, the left-side $J_{t_1}^{-(Row,\,t_2)}(\mathbf{x}_{1t_2},..., \mathbf{x}_{t_1t_2})$ and the right-side one $J_{t_1}^{+(Row,\,t_2)}(\mathbf{x}_{t_1t_2},..., \mathbf{x}_{N_1t_2})$, which differ from (33) by that the summation in both sums is fulfilled, respectively, only up to the t_1 th summand and starting from it. Functions

$$\tilde{J}_{t_1}^{-(Row,\,t_2)}(\mathbf{x}_{t_1t_2}) = \min_{\mathbf{x}_{s_1t_2}:\, s_1<t_1} J_{t_1}^{-(Row,\,t_2)}(\mathbf{x}_{1t_2},..., \mathbf{x}_{t_1t_2}),$$

$$\tilde{J}_{t_1}^{+(Row,\,t_2)}(\mathbf{x}_{t_1t_2}) = \min_{\mathbf{x}_{s_1t_2}:\, s_1>t_1} J_{t_1}^{+(Row,\,t_2)}(\mathbf{x}_{1t_2},..., \mathbf{x}_{t_1t_2})$$

are, respectively, the left-side and the right-side Bellman functions. These functions can be recurrently found in each row t_2 by the dynamic programming procedure "forward and counter-forward" (Section 2.3). Then the marginal functions $\hat{J}_{t_1}^{(Row,\,t_2)}(\mathbf{x}_{t_1t_2})$ can be easily found in accordance with (15) in Theorem **2.**

Thus, two independent dynamic programming procedures, the left-side and the right-side one, when applied independently to each row of the pixel grid, give the marginal functions $\hat{J}_{t_1}^{(Row,\,t_2)}(\mathbf{x}_{t_1t_2})$, $\mathbf{x}_{t_1t_2} \in \mathsf{F}$, at all pixels. After that, two runs of the dynamic programming procedure (forward and back), applied independently to each column of the pixel grid, result in the displacement vector field (32) of the sought elastic transformation found from (35).

Fig. 6 shows three examples of matching different shots of the same person for three kinds of distinctions in shooting conditions.

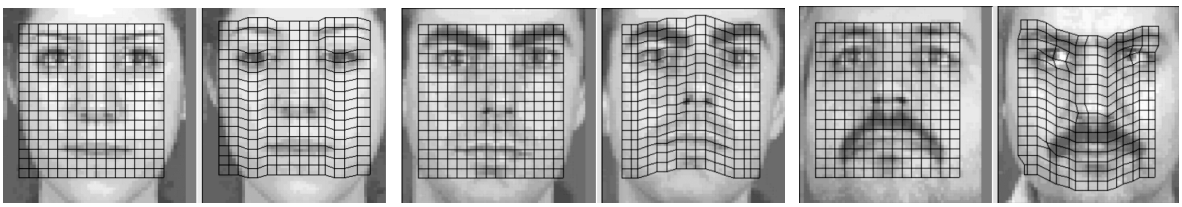


Figure 6. Effect of elastic transformation of the image pixel grid in the case of differences (from left to right) in facial expression, camera angle and both camera angle and facial expression.

The computational complexity of the dynamic programming procedure applied to all rows (columns) of the reference-image interior T' is proportional to the number of pixels $|T'|$. One step of that procedure has computational complexity proportional, theoretically, to the squared

number of elements $|F|^2$ in the set of admitted displacement vectors $\mathbf{x_t} = (x_{\mathbf{t}}^{(1)}, x_{\mathbf{t}}^{(2)}) \in F$ (28), but this coefficient can be reduced to $|F|$ if we take into account that the unsmoothness penalty in (29) increases drastically with the growth of $x_{\mathbf{t}}^{(1)}$ and $x_{\mathbf{t}}^{(2)}$. All in all, the computational complexity of the above-described matching algorithm turns out to be proportional to the product $|T'||F|$.

It should be marked that the dynamic programming procedure is applied independently to each of N_2 rows and then to each of N_1 columns of the pixel grid, therefore, it can be implemented on a parallel computer. In this case, the time of calculations will be proportional to the sum $N_1 + N_2$ instead of the product $|T'| = N_1 N_2$.

5 Support object classifiers with elastic kernel functions for face identification

5.1 Structure of support-object decision rules of face identification

One of the aims of this work is to examine the applicability of the support vector method [18] combined with elastic transformation if the image pixel grid to the problem of face identification. In accordance with the support vector method, the class membership of a new object Y is determined on the basis of evaluating its similarity to few selected objects of the training set Y_j, $j = 1,...,M$ by computing a pre-specified kernel function $K(Y, Y_j)$. Thus, following R. Duin [17], we call them here support objects instead of Vapnik's support vectors.

The discriminant function that would accomplish the recognition of shots of a specific client $g_j = 1$ against all the others $g_j = -1$ can be inferred from the training set $\{Y_j, j = 1,..., M\}$ by the support object technique with a kernel function $K(Y', Y'')$ of that or other kind. Such a discriminant function, having the structure

$$d(Y) = \sum_{j=1}^{M} a_j K(Y, Y_j) + b \gtrless 0, \tag{36}$$

can be found by solving the quadratic programming problem [18]

$$\begin{aligned} &\sum_{j=1}^{M}\sum_{k=1}^{M} K(Y_j, Y_k) a_j a_k + C\sum_{j=1}^{M} \delta_j \to \min, \\ &g_j\left(\sum_{k=1}^{M} a_k K(Y_j, Y_k) + b\right) \geq 1 - \delta_j, \quad j = 1,..., M. \end{aligned} \tag{37}$$

It is particularly convenient to solve this problem in the dual form, which is also a quadratic programming problem

$$\begin{aligned} &2\sum_{j=1}^{M} \lambda_j - \sum_{j=1}^{M}\sum_{k=1}^{M} g_j g_k K(Y_j, Y_k) \lambda_j \lambda_k \to \max, \\ &\sum_{j=1}^{M} g_j \lambda_j = 0, \quad 0 \leq \lambda_j \leq C/2, \quad j = 1,..., M, \end{aligned}$$

whose arguments are Lagrange multipliers $\lambda_j \geq 0$ at the constraints in (37), i.e. at vectors Y representing the training faces. Those of the optimal values of Lagrange multipliers which turn out to be positive determine the nonzero coefficients of the optimal discriminant function $a_j = \lambda_j g_j$. The criterion (37) leads to the constant b that provides the symmetric position of the hypersurface $d(Y) = 0$ between the subsets of two classes in the training set

$$b = \frac{1}{2}\left[\min_{j:\, g_j=1} \sum_{k=1}^{M} a_k K(Y_j, Y_k) - \max_{j:\, g_j=-1} \sum_{k=1}^{M} a_k K(Y_j, Y_k) \right]. \tag{38}$$

As an alternative to the symmetric choice of constant b in (38), we tried also another way of finding this parameter [23] which is based on the fact that the error percentage calculated in the training set by the leave-one-out procedure is an almost unbiased estimate of the error probability that a discriminant function built by the technique under consideration will show in the general population [18]. Therefore, it appears reasonable to choose such a value of parameter b that provides, in the leave-one-out procedure, the minimum percentage of errors.

Let $(a_j^{(k)},\ j = 1,..., M,\ j \neq k)$ be the coefficients of the discriminant function inferred from $N-1$ objects of the training set without the k th object, and $D^{(k)} = \sum_{j=1,\ j\neq k}^{M} a_j^{(k)} K(Y_k, Y_j)$. Parameter b is determined from the condition

$$b = \arg\min E(z),$$

$$E(z) = \left\{ \frac{1}{M^1} \sum_{j=1,\ g_j=1}^{M} I[D^{(k)} < z] + \frac{1}{M^{-1}} \sum_{j=1,\ g_j=-1}^{M} I[D^{(k)} > z] \right\}, \tag{39}$$

where $I[\cdots]$ is an indicator functions that equals 1 when the condition in the brackets is satisfied and 0 otherwise, $M^1 + M^{-1} = M$ are the numbers of objects of the first and second class. It is easy to see that such a criterion provides the minimum error in classifying the left-out objects by discriminant functions of type (36).

5.2 Rigid and elastic kernel functions

The inner product of two gray-level vectors

$$K(Y', Y'') = (Y', Y'') = (Y'^T Y'') = \sum_{\mathbf{t} \in T'} y'_{\mathbf{t}} y''_{\mathbf{t}} \tag{40}$$

is nothing else than a kernel function which we call linear because it leads to the family of linear discriminant functions in the feature vector space. However, other kinds of kernel functions are also traditionally considered [18], such as the parametric families of polynomial

$$K(Y', Y'') = \{(Y', Y'') + 1\}^{\alpha} \tag{41}$$

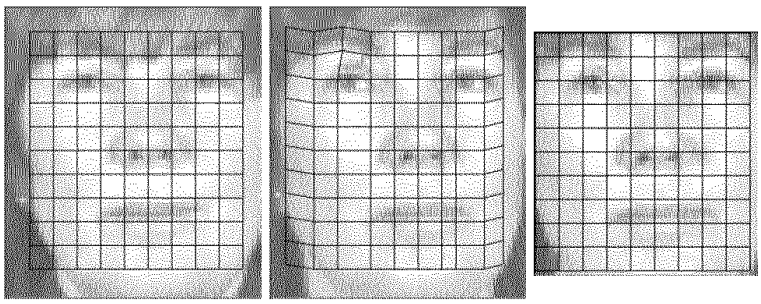
and radial kernel functions

$$K(Y', Y'') = \exp\{-\alpha \| Y' - Y'' \|^2\} = \exp\{-\alpha[(Y', Y') + (Y'', Y'') - 2(Y', Y'')]\}. \tag{42}$$

In this work, the kernel functions like (40), (41) and (42) are called rigid because the inner products in them are formed by identical pixels of images to be compared. The idea of an elastic transformation of, at least, one of two image planes before calculating the inner product of images in order to equalize the positions of identical points in both faces (Fig. 7) follows from the intent to compensate inevitable distortions caused by different camera angles or different facial expressions:

$$B(Y', Y'') = \sum_{\mathbf{t} \in T'} y'_{\mathbf{t}} y''_{\mathbf{t}+\mathbf{x}_{\mathbf{t}}} \ . \tag{43}$$

Symbol $B(Y',Y'')$ is used here to emphasize that it is just brightness of both images that is being compared here. It is this idea which underlies the notion of elastic kernel functions we consider here as a modification of the traditional approach.



(a) (b) (c)

Figure 7. Two shots Y' (a) and Y'' (b) of the same face taken under different camera angles and the result of matching the reference shot (a) to the probe (b). Image (c) is obtained by "straightening" the conventional reference net in the probe and demonstrates the idea of elastic kernel function.

The idea of kernel function (40), (41) or (42) with elastic inner product (43) originates from the expectation that brightness pictures of two shots of the same face may be approximately obtained from each other by an elastic transformation of the image plane. Therefore, we call kernel functions

$$K_B(Y',Y'') = B(Y',Y''), \tag{44}$$

$$K_B(Y',Y'') = \{B(Y',Y'')+1\}^{\alpha}, \tag{45}$$

$$K_B(Y',Y'') = \exp\{-\alpha[B(Y',Y')+B(Y'',Y'')-2B(Y',Y'')]\}, \tag{46}$$

that fulfill brightness comparison (43), the brightness-based elastic kernel functions.

On the other hand, it is also to be expected that a lesser distortion of the image plane will be required for achieving the minimum value of the criterion (29) when shots of the same face are compared than in the case of different faces. The transformation intensity may be measured by norms $\|\mathbf{x}_\mathbf{t}\|$ of shift vectors making the vector field $X = (\mathbf{x}_\mathbf{t}, \mathbf{t} \in T')$ that attains the minimum point of criterion (29).

Besides the brightness-based kernel functions, we consider here also another kind of elastic kernel functions $K_T(Y',Y'')$, called transformation-based, in which, instead of elastic inner product of image intensities (43), the images are compared by the intensity of the elastic transformation between them

$$T(Y',Y'') = \sum_{\mathbf{t}\in T'} \|\mathbf{x}_\mathbf{t} - \bar{\mathbf{x}}\|^2,$$

where $\bar{\mathbf{x}} = (1/|T'|)\sum_{\mathbf{s}\in T'}\mathbf{x}_\mathbf{s}$ is the average value of shift vectors within the inner part of the reference image pixel grid T' that consists of $|T'|$ pixels. Subtracting this value before measuring the norms of shift vectors pursues the aim of preventing the penalization of parallel displacement of the whole image.

The measure $T(Y',Y'')$ has the meaning of squared distance between images, thus, it is best suited for the radial transformation-based elastic kernel function (42)

$$K_T(Y',Y'') = \exp\{-\alpha T(Y',Y'')\}. \quad (47)$$

The brightness-based and the transformation-based elastic kernel functions, respectively, (44)-(46) and (47), assess the image similarity from two different points of view. A natural way of fusing these two kinds of information is to form a combined elastic kernel function as a linear combination of the brightness-based and the transformation-based ones:

$$K_{B,T}(Y',Y'') = K_B(Y',Y'') + \delta K_T(Y',Y''). \quad (48)$$

The positive coefficient δ is required here to equalize the difference in scales caused by the different structure of the two kinds of kernel functions.

5.3 Data base and experimental results

The experiments were carried using the XM2VTS face data base recorded at the University of Surrey [24]. We used the experimental set consisting of 2360 face shots of 295 persons (8 shots per person) that were conventionally divided into 200 clients and 95 impostors. Among the 8 shots of each client, 6 were chosen as the training set, whereas the remaining 2 shots were considered as belonging to the test set. The set of impostors was also divided into the training and the test set that contained, respectively, 25 and 70 persons with 8 shots per person. In each experiment on face verification, the training set was made up by 6 shots of the respective client and 1394 shots of other people, whereas the test set consisted of 2 shots of this person and 560 shots of test impostors. All in all, we formed 200 training sets and, respectively, 200 test sets, in accordance with the number of clients represented in the data base.

Rigid kernel functions. For the linear rigid kernel function (40), we inferred two discriminant functions for face verification defined in (36) from each of the 200 training sets, i.e. for each of the 200 clients, using, respectively, the symmetric (38) and asymmetric constant b (39). In each of the 400 experiments, we estimated the error rate on the test set.

For the polynomial and radial rigid kernel functions (41) and (42), we repeated the procedure, but the training was initiated several times to find the optimal value of parameter α as measured by the criterion of minimum error rate estimated using the lave-one-out procedure.

The error rates achieved on the test set are presented in the following table:

	Kernel (40)	Kernel (41) (optim. α)	Kernel (42) (optim. α)
Symmetric (38)	27.0%	24.5%	27.0%
Asymmetric (39)	5.7%	5.0%	5.1%

Brightness-based elastic kernel functions. The experiments were performed using the same scheme as with the rigid kernel functions. The following table shows the final error rates estimated on the test set:

	Kernel (44)	Kernel (45) (optim. α)	Kernel (46) (optim. α)
Symmetric (38)	13.2%	13.5%	13.4%
Asymmetric (39)	3.2%	1.6%	1.8%

As can be seen, the asymmetric choice of constant b gave better results for all kinds of kernel functions, the rigid as well as the elastic ones. All in all, the polynomial elastic kernel (45) gave the best performance.

Combined elastic kernel function. In this series of experiments, we tried the combinations (48) of three kinds of brightness-based elastic kernels (44), (45) and (46) with the same transformation-based kernel (47). In each experiment, we used the asymmetric version of constant b in the discriminant function, and took the same optimal value of parameter α found, respectively, for kernels (45) and (46) in the previous series. Instead, we varied coefficient δ to find the optimal value in (48). We obtained the following results:

	Kernel (44)	Kernel (45)	Kernel (46)
Asymmetric (39)	1.38%	1.56%	2.1%

Discussion. As we see, the employment of an elastic kernel function instead of traditional rigid ones essentially improves the efficiency of face verification. The superior performance of the combined elastic kernel function built as a combination of one of brightness-based kernels and the only transformation-based one we have considered here is an evidence in favour of the hypothesis that information on the identity of a person contained in the face image is spread between brightness and form.

6 Conclusions

We have proposed to apply the idea of measuring the face image similarity by elastic graph matching to the original pixel neighborhood graph with single brightness values at its nodes as a trivial graph-based model of the reference image. In this case, there is no need to extract fiducial points of the face image and evaluate local features of brightness distribution in some windows around those points. So, there is no danger that the corresponding feature vectors in the two images will be incomparable because the required deformation of the image plane is not yet known.

To overcome the dramatically increasing number of variables in the lattice-like pair-wise separable objective function of elastic graph matching, we have replaced it by a series of partial tree-like ones, the minimization of each of which by a fast dynamic programming algorithm gives the sought displacement vectors in a single column of the reference-image pixel grid.

The same optimization technique may be used also for elastic matching of traditional graph-based models with the lattice of nodes at selected fiducial points of the face image.

We have shown that the employment of an elastic kernel function instead of traditional rigid ones essentially improves the efficiency of face verification. The superior performance of the combined elastic kernel function built as a combination of one of brightness-based kernels and the only transformation-based one we have considered here is an evidence in favour of the hypothesis that information on the identity of a person contained in the face image is spread between brightness and form. Thus, it appears reasonable to study the transformation-based way of building elastic kernels more carefully. From the mathematical point of view, the combination of several kernels by summing them is completely equivalent to extending the feature space by adding new features. s that information on the identity of a person contained in the face image is spread between brightness and form.

7 Acknowledgements

This work was done in the period January 15 – March 2, 2002, when Dr. Vadim Mottl visited the Rutgers University on the invitation of Director of DIMACS Professor Fred Roberts. The authors are grateful to him for this invitation.

The authors would like to thank Professor Josef Kittler, Head of the Vision, Speech and Signal Processing Group at Surrey University, Great Britain, for the possibility to work with the XM2VTS face data base, his interest to this work, and extremely helpful advice.

References

1. Muchnik I., Mottl V. Bellman Functions on Trees for Segmentation, generalized Smoothing and Multi-Alignment in Massive Data Sets. DIMACS Technical Report 98-15, February, 1998.
2. Muchnik I., Mottl V., Levyant V. Massive Data Set Analysis in Seismic Explorations for Oil and Gas in Crystalline Basement Interval. DIMACS Technical Report 99-03, January, 1998.
3. Mottl V., Blinov A., Zabusky N., Muchnik I. Apparent Motion in Spatial Time-varying Data: A Variational Approach to Pointwise Tracking of Coherent Structures in Computational Fluid Dynamics. DIMACS Technical Report 99-55, November, 1999.
4. Mottl V., Blinov A., Kopylov A., Kostin A., Muchnik I. Variational methods in signal and image analysis. 14th International Conference on Pattern Recognition. Brisbane, Australia, August 16-20, 1998, Vol. 1.
5. Bellman, R. Dynamic Programming. Princeton University Press, Princeton, N.J., 1957.
6. Sniedovich M. Dynamic Programming. Marcel Dekker, NY, 1991.
7. Bertele U., Brioshi F. Nonserial Dynamic programming. Academic Press, New York and London, 1972.
8. Berge C. Graphs and Hypergraphs. New York: Elsevier, 1973.
9. Berge C. Hypergraphs: The Theory of Finite Sets. Amsterdam, Netherlands: North-Holland, 1989.
10. Kirkpatrick S., Gelatt C.D., Jr., Vecchi M.P. Ottimization by simulated annealing. Science, 1983, 220, pp. 671-680.
11. Geman S., Geman D. Stochastic relaxation, Gibbs distributions, and the beyesian restoration of images. IEEE Trans. on PAMI, 1984, 6, pp. 721-741.
12. Geman S., Graffine C. Markov random field image models and their applications to computer vision. Proceedings of the 1986 International Congress of mathematicians (A.M. Gleason, ed.). American Mathematical Society, R.I., 2, pp. 1496-1517.
13. Mottl V.V., Muchnik I.B., Blinov A.B., Kopylov A.V. Hidden tree-like quasi-Markov model and generalized technique of for a class of image processing problems. 13th International Conference on Pattern Recognition. Vienna, Austria, August 25-29, 1996. Track B, pp. 715-719
14. Chow C.K., Liu C.N. Approximating discrete probability distributions with dependence trees. IEEE Trans. on Inf. Theory, 1968, IT-14, 3, pp. 162-167.

15. Wu C.-H., Doershuk P.C. Tree approximation to Markov random fields. IEEE Trans. on PAMI, 1995, Vol. 17, No. 4, pp. 391-402.
16. Gong S., McKenna S.J., Psarrou A. Dynamiv Vision: From Images to Face Recognition (Image Processing). Imperial College Press, 2000.
17. Duin R.P.W., Pekalska E., De Ridder D. Relational discriminant analysis. Pattern Recognition Letters, 1999, Vol. 20, No. 11-13, pp. 1175-1181.
18. Vapnik V. Statistical Learning Theory. John-Wiley & Sons, Inc., 1998.
19. Lades M., Vorbrügen J., Buhman J., Lange J., Malsburg C., Würtz R., Kohnen W. Distortion invariant object recognition in dynamic linc architecture. IEEE Transactions on Computers, 1993, Vol. 42, No. 3, pp. 300-311.
20. Mottl V., Kopylov A., Kostin A., Yermakov A., J. Kittler. Elastic transformation of the image pixel grid for similarity based face identification. Submitted to the 16th International Conference on Pattern Recognition, Quebec City, Canada, August 11-15, 2002.
21. Kostin A., Kopylov A., Mottl V., Muchnik I. Dynamic programming procedures in non-stationary signal analysis. Pattern Recognition and Image Analysis. Advances in Mathematical Theory and Applications. 2001, Vol. 11, No. 1, pp. 205-208.
22. Chellappa R., Wilson C.L., Sirohey S. Human and machine recognition of faces: A survey. Proceedings of the IEEE, 1995, 83, No. 5, pp. 704-740.
23. Seredin O. Methods and Algorithms of Featureless Pattern Recognition in Hilbert Space. Ph.D. Thesis. Tula State University, Tula, Russia, 2001.
24. Jonsson K.T. Robust Correlation and Support Vector Machines for Face Identification. Ph.D. Thesis. University of Surrey, Guildford, United Kingdom, 2000.