A MULTIGRID APPROACH FOR MULTI-DIMENSIONAL SCALING*

MICHAEL M. BRONSTEIN , ALEXANDER M. BRONSTEIN , RON KIMMEL , AND IRAD YAVNEH†

Abstract.
A multigrid approach for the efficient solution of large-scale multidimensional scaling (MDS) problems is presented. The main motivation is a recent application of MDS to isometry-invariant representation of surfaces, in particular, for expression-invariant recognition of human faces. Simulation results show that the proposed approach significantly outperforms conventional MDS algorithms.

Key words. multigrid, multiresolution, isometric embedding, multidimensional scaling, SMACOF, face recognition, bending-invariant canonical form, dimensionality reduction.

AMS subject classifications. 91C15

1. Introduction. *Multidimensional scaling* (MDS) is a generic name for a family of algorithms that construct a configuration of points in a target metric space from information about inter-point distances (*dissimilarities*), measured in some other metric space. The range of MDS applications is very broad and ranges from stock market analysis [18] to computational chemistry [1] and breast cancer diagnosis [23]. MDS is widely used in dimensionality reduction, data analysis and visualization applications, when, for example, one wishes to understand complicated high-dimensional data structures and represent them by low-dimensional ones [3, 24].

From the point of view of the underlying optimization problem, MDS is known to be hard, as it involves a nonlinear non-convex objective function requiring heavy computation, whose gradient and Hessian are also heavy to compute, and moreover, the Hessian is full. Current MDS algorithms are notoriously slow, and limited to small data sets. Efficiently solving large-scale MDS problems arising in numerous applications has been a challenge for long time.

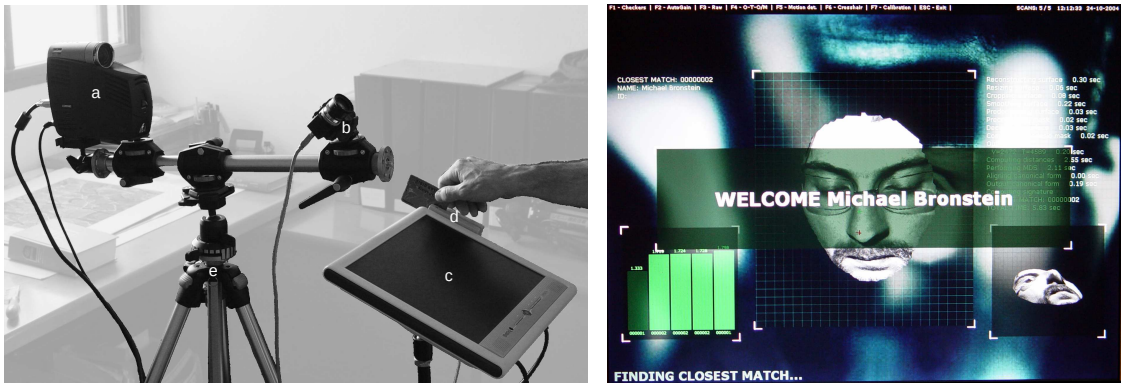
In many cases, the dissimilarities can be thought of as geodesic distances measured on a smooth Riemannian manifold, and the underlying geometry can be used to advantage. This particular setting of the MDS problem is known as the *isometric embedding* problem, and from the geometric point of view, it is the problem of representing the intrinsic structure of a Riemannian manifold in some other metric space. The isometric embedding problem was found to be important in numerous applications. Schwartz et al. [26] embedded a convoluted 3D surface of the brain cortex into a plane to study its structure as a 2D image. A similar approach was used in [28, 19] for texture mapping. Elad and Kimmel [14, 16] proposed embedding surfaces into a higher dimensional Euclidean space in order to compute their isometry-invariant signatures, which were called *canonical forms*. This approach was applied in [7, 8] to the problem of 3D face recognition.

The results presented here are motivated mainly by the 3D face recognition application. One of the greatest challenges in human face recognition is the ability to recognize people with varying facial expressions, as facial expressions can change the facial appearance dramatically. The 3DFACE prototype 3D face recognition system developed at the Department of Computer Science, Technion (Figure 1.1) uses MDS to compute an expression-invariant representation (canonical form) of the facial surface [8]. Constructing such a representation requires an efficient solution to a large-scale isometric embedding problem. One of the most crucial considerations in the face recognition application is the computation time. In the current implementation, near real-time computation of canonical forms is achieved mainly by CPU power and "technically" rather than algorithmically fast implementation in C, optimized for AMD Opteron 64-bit processor with SSE extensions [10].

In this paper, we propose the multigrid framework to boost the performance of standard MDS algorithms. The scope of the paper is the following: In Section 2, we define the MDS problem and standard methods used for its solution. Section 3 describes the particular instance of the MDS problem

*This work was supported in part by the Israel Science Foundation (ISF), Grant No. 738/04 and Bar Nir Bergreen Software Technology Center of Excellence (STL).

†Department of Computer Science, Technion – Israel Institute of Technology, Haifa 32000, Israel (bronstein@ieee.org, alexbron@ieee.org, ron@cs.technion.ac.il, irad@cs.technion.ac.il).



(a) Face recognition system (b) User interface

FIG. 1.1. *A prototype 3D face recognition system developed in the Technion (a) and a screenshot from its user interface (b) showing correct recognition of one of the authors.*

arising in isometric embedding of Riemannian manifolds. In Section 4 we introduce a multigrid MDS algorithm. Section 5 shows experimental results on the problem of facial surface embedding, and Section 6 concludes the paper.

2. Multidimensional scaling. Let $\mathbf{\Delta}$ be a symmetric $N \times N$ dissimilarity matrix with non-negative elements δ_{ij} and zero diagonal. The goal of multidimensional scaling is to find a set of points $\mathbf{x}_1, ..., \mathbf{x}_N$ in an m-dimensional metric space with the metric $d_{ij} \equiv d(\mathbf{x}_i, \mathbf{x}_j)$, such that $d_{ij} \approx \delta_{ij}$ for all $i, j = 1, ...N$. In the majority of cases, the target space is an m-dimensional Euclidean space $\mathbb{R}^m$ (which is also the assumption here), though other choices are also possible [15, 9, 27]. Practically, this influences only the way in which the distances d_{ij} are computed.

One of the common methods in MDS is to obtain the configuration of points by the minimization of the *stress*

$$s(\mathbf{X}; \mathbf{\Delta}, \mathbf{W}) = \sum_{i>j} w_{ij}(d_{ij}(\mathbf{X}) - \delta_{ij})^2, \tag{2.1}$$

where $\mathbf{X} = (x_{ij})$ is the $N \times m$ matrix of coordinates of the resulting points in an m-dimensional space, called the *configuration matrix* [21]. The symmetric $N \times N$ matrix of weights $\mathbf{W} = (w_{ij})$ determines the relative contribution of distances to the error criterion. We will distinguish between the *weighted stress*, where some values of w_{ij} are specified, and the *non-weighted stress*, where $w_{ij} = 1$.

2.1. First-order methods. Optimization of $s(\mathbf{X})$ can be performed by first-order, gradient descent-type methods, in which the direction at the $k+1$st iteration is $\mathbf{X}^{(k+1)} = -\nabla s(\mathbf{X}^{(k)})$. The gradient of $s(\mathbf{X})$ with respect to $\mathbf{X}$ is given by

$$\frac{\partial}{\partial x_{kl}} s(\mathbf{X}; \mathbf{\Delta}, \mathbf{W}) = 2 \sum_{j \neq k} w_{kj} \frac{(d_{kj} - \delta_{kj})}{d_{kj}} (x_{kl} - x_{jl}), \quad k = 1, ..., N; l = 1, ..., m; \tag{2.2}$$

and can be written as

$$\nabla s(\mathbf{X}; \mathbf{\Delta}, \mathbf{W}) = 2\mathbf{V}\mathbf{X} - 2\mathbf{B}(\mathbf{X}; \mathbf{\Delta}, \mathbf{W})\mathbf{X}, \tag{2.3}$$

where $\mathbf{V}$ is a matrix with elements

$$v_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j \\ \sum_{j=1}^{N} w_{ij} & \text{if } i = j \end{cases}, \tag{2.4}$$

and $\mathbf{B}$ is a matrix with elements

$$b_{ij} = \begin{cases} -w_{ij}\delta_{ij}d_{ij}^{-1}(\mathbf{X}) & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) \neq 0 \\ 0 & \text{if } i \neq j \text{ and } d_{ij}(\mathbf{X}) = 0 \\ -\sum_{j \neq i} b_{ij} & \text{if } i = j \end{cases} . \tag{2.5}$$

It was observed [20] that the first-order optimality condition, $\nabla s(\mathbf{X}) = 0$, can be written as $\mathbf{X} = \mathbf{V}^\dagger \mathbf{B}(\mathbf{X})\mathbf{X}$, and that the sequence

$$\mathbf{X}^{(k+1)} = \mathbf{V}^\dagger \mathbf{B}(\mathbf{X}^{(k)})\mathbf{X}^{(k)}, \tag{2.6}$$

converges to the local minimum of $s(\mathbf{X})$ (here $\dagger$ denotes matrix pseudoinverse). The algorithm using this multiplicative update is called SMACOF [12, 13, 3]. It can be easily shown to be equivalent to weighted gradient descent with constant step size $\mathbf{X}^{(k+1)} = -\frac{1}{2}\mathbf{V}^\dagger \nabla s(\mathbf{X}^{(k)})$, and if a non-weighted stress is used, it is essentially a gradient descent with constant step size $\mathbf{X}^{(k+1)} = -\frac{1}{2N}\nabla s(\mathbf{X}^{(k)})$.

SMACOF is widely used for large-scale MDS problems. Its disadvantage is slow convergence in the proximity of the minimum, which is inherent to all first-order methods.

2.2. Second-order methods. Some recent works (e.g. [21]) propose using second-order (Newton-type) algorithms for stress minimization. A basic Newton iteration has the form $\mathbf{X}^{(k+1)} = -\mathcal{H}^{-1}\nabla s(\mathbf{X}^{(k)})$, where $\mathcal{H}$ represents the Hessian, which is a fourth-order tensor in this notation. If $\mathbf{X}$ is column-stacked into a $Nm \times 1$ vector, the Hessian can be represented by a $Nm \times Nm$ matrix, consisting of m^2 blocks of size $N \times N$. Each block is given by

$$\mathbf{H}^{kl} = \begin{cases} \tilde{\mathbf{H}}^{kl} & \text{if } k \neq l \\ \tilde{\mathbf{H}}^{kl} + 2(\mathbf{V} - \mathbf{B}) & \text{if } k = l \end{cases} ; \quad 1 \leq k, l \leq m, \tag{2.7}$$

where

$$\tilde{h}_{ij}^{kl} = \begin{cases} -2w_{ij}(x_{ik} - x_{jk})(x_{il} - x_{jl})\delta_{ij}d_{ij}^{-3} & \text{if } i \neq j \\ 2\sum_j w_{ij}(x_{ik} - x_{jk})(x_{il} - x_{jl})\delta_{ij}d_{ij}^{-3} & \text{if } i = j \end{cases} . \tag{2.8}$$

The Newton method has quadratic convergence, as, in the proximity of the minimum, a function is accurately approximated by a quadratic function. The disadvantage of the Newton method is the relatively high computational complexity required for the Hessian construction and inversion.

2.3. Remarks. The MDS problem is in all respects a hard optimization problem. The stress is a nonlinear non-convex function w.r.t. $\mathbf{X}$, therefore convex optimization algorithms do not guarantee convergence to a global minimizer of $s(\mathbf{X})$ and can converge to local minima. Practice shows, however, that often a good initialization can prevent convergence to local minima; we will show a way to choose such an initialization in the particular case of isometric embedding.

Another characteristic of the MDS problem is the high computational complexity of the stress function and its derivatives; this stems from the fact that the matrix $\boldsymbol{\Delta}$ is full. Analyzing the computational complexity of the stress $s(\mathbf{X})$ and its derivatives $\nabla s(\mathbf{X}), \nabla^2 s(\mathbf{X})$ (see Table 2.1), two conclusions can be made. First, the stress and the gradient computation complexity is roughly the same. It therefore follows that the complexity of line search, which is necessary in first-order algorithms like conjugate gradients, becomes excessive. In many cases, gradient descent with constant step size (or, equivalently, the SMACOF algorithm with multiplicative update) is advantageous. Secondly, the Newton algorithm is efficient only for small-scale problems (practically, hundreds of points), as the cost of Hessian construction and inversion (requiring $\mathcal{O}(N^3m^3)$ operations) becomes prohibitive with large N.

Unfortunately, though the Hessian is structured, it is *not* sparse. The diagonals of each block are N times larger than the rest of the elements, which leads to a structure with dominant $2m - 1$ diagonals. The Hessian can be approximated by such a multidiagonal matrix, yet, since the diagonals in each block consist of sums of all the block elements, the construction cost of the sparse Hessian

TABLE 2.1
Approximate computational complexity (in terms of multiplication operations) of the stress function $s(\mathbf{X})$ and its derivatives $\nabla s(\mathbf{X}), \nabla^2 s(\mathbf{X})$. N denotes the number of points, m denotes the embedding space dimension. C_s denotes the complexity of square root computation in terms of multiplication operations; typically $C_s \approx 25$.

	Weighted	Non-weighted
s	$\frac{1}{2}N(N-1)(3+C_s)$	$\frac{1}{2}N(N-1)(2+C_s)$
∇s	$\frac{1}{2}N(N-1)(3+C_s)+2Nm$	$\frac{1}{2}N(N-1)(2+C_s)+Nm$
$\nabla^2 s$	$\frac{1}{2}N(N-1)(8+C_s)m^2$	$\frac{1}{2}N(N-1)(7+C_s)m^2$

approximation is similar to the construction cost of the full Hessian. The only advantage that can be achieved by this is the reduction of the Hessian inversion complexity for each Newton step. However, the convergence of the Newton method using such an approximation is usually inferior in comparison to other alternatives.

3. Isometric embedding. In a general MDS problem, the only data given is the dissimilarity matrix $\mathbf{\Delta}$, and it can arise from any finite metric space. Usually, there is no additional information about the "origin" of these dissimilarities. Yet, in many cases it can be assumed that the dissimilarities are geodesic distances on a smooth Riemannian manifold. We call such cases the isometric embedding problem. In isometric embedding, we represent the intrinsic geometry of a smooth Riemannian manifold using the intrinsic geometry of some other manifold (in our case, $\mathbb{R}^m$).

As an illustration, think of the problem of Earth mapping (also known as the "map-maker problem"). We wish to map the spherical surface of the Earth into a plane, preserving in the best possible way the distances between geographic objects (Figure 3.1). In other words, we embed a sphere into an Euclidean space. Since a "pure" isometric embedding does not exist in this case, we are looking for a near-isometric embedding that minimally distorts the original geodesic distances.

In the discrete setting, we have an m'-dimensional manifold $\mathcal{S}$, which is sampled at N points $\xi_1, ..., \xi_N$, and the geodesic distances $\delta_{ij} \equiv \delta(\xi_i, \xi_j)$ between all the points are computed. Isometric embedding is a mapping between two finite metric spaces,

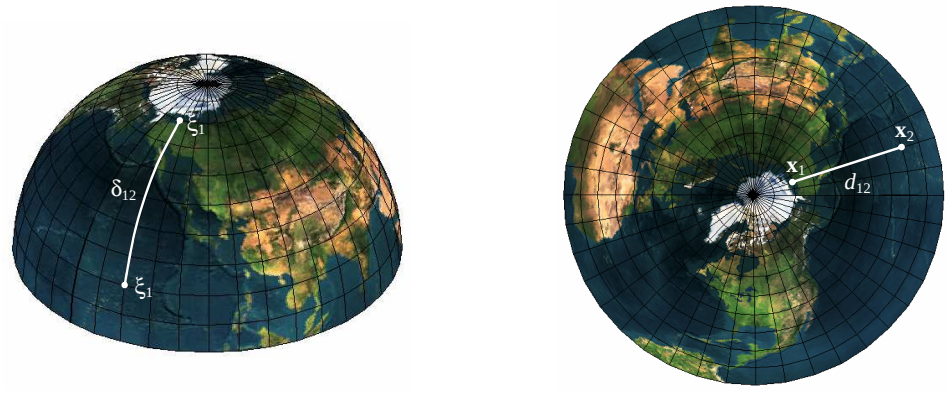
$$\varphi : (\{\xi_1, ...\xi_N\} \subset \mathcal{S}, \mathbf{\Delta}) \to (\{\mathbf{x}_1, ...\mathbf{x}_N\} \subset \mathbb{R}^m, \mathbf{D}), \tag{3.1}$$

trying to achieve $d_{ij} \approx \delta_{ij}$. This can be seen as a particular case of the general MDS problem, in which we assume that the dissimilarities δ_{ij} arise from sampling the geodesic distances on a smooth Riemannian manifold. What is more important, is that the geometric relations between the original points are known. They can either be given explicitly, as a grid or triangulation, or inferred from $\mathbf{\Delta}$ itself.[1]

In the 3D face recognition application, which was our main motivation, isometric embedding is used to create expression-invariant representations of facial surfaces [7]. The embedding allows to get rid of the deformations of the facial surface introduced by the facial expressions (which can be approximated as isometries of the facial surface [11]) and creates a representation which can be treated as a rigid object (see example in Figure 3.2). The facial surfaces (two-dimensional manifolds) are represented by $N \approx 3000$ points and N^2 corresponding inter-point geodesic distances, which are measured numerically using the Fast Marching Algorithm [22]. Then, 40 iterations of the SMACOF algorithm are performed to embed the facial surface into $\mathbb{R}^3$; as the initialization, the original 3D coordinates of the points are used (this allows to avoid convergence to local minima in most cases [10]).

4. Multigrid isometric embedding. Our previous results [10] demonstrated that substantial performance improvement can be achieved by using a multiresolution initialization to the embedding problem. Here, we extend those attempts into a genuine multigrid approach.

[1] One possibility is to define adjacent points as nearest neighbors in the sense of the dissimilarity δ_{ij}. For example, starting with a point 1, we find its nearest neighbor $i_1 = \mathrm{argmin}_j \delta_{1j}$. These two points will be merged at a coarser resolution level, with the dissimilarities δ_{ij} involved in the computation of the interpolation coefficients (e.g. interpolation coefficients inversely proportional to δ_{ij} or δ_{ij}^2). The process is repeated for the remaining points in a similar manner. This idea is general and not limited to smooth Riemannian manifolds.



(a) Original surface (b) Embedding into $\mathbb{R}^2$

FIG. 3.1. *Illustration of the isometric embedding problem arising in cartography. A planar map (b) of the Earth (a) is created by embedding the spherical surface into* $\mathbb{R}^3$*, such that the geodesic distances are replaced with Euclidean ones.*

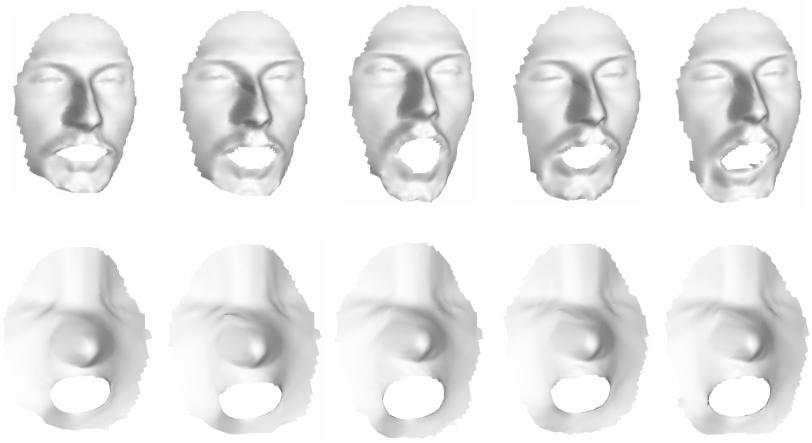


FIG. 3.2. *Example of canonical forms (second row) of facial surfaces (first row) and their insensitivity to facial expressions.*

Originally, multigrid methods were introduced in the context of differential equations [17, 2, 4, 5]. More recently, this framework was adapted to non-linear discrete optimization problems (see for example [25]). The optimization problem $\min_{\mathbf{X}} s(\mathbf{X})$ is equivalent to the solution of the non-linear equation $\nabla s(\mathbf{X}) = 0$, arising from the first-order optimality conditions. The spirit of multigrid is to solve the nonlinear problem $\nabla s(\mathbf{X}) = 0$ using a sequence of approximate solutions to nonlinear problems of the form $\nabla s(\mathbf{X}) = \mathbf{T}$, solved on coarse grids. The term $\mathbf{T}$ arises from the residual transferred from previous levels. In the optimization problem formulation, we need to minimize functions of the form $s(\mathbf{X}) - \mathrm{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{T})$, whose gradient equals $\nabla s(\mathbf{X}) - \mathbf{T}$.

4.1. Modified stress. The second (linear) term makes the function $s(\mathbf{X}) - \mathrm{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{T})$ unbounded. To overcome this problem, we define the *modified stress*

$$\hat{s}(\mathbf{X}; \boldsymbol{\Delta}, \mathbf{W}) = \sum_{i>j} w_{ij}(d_{ij}(\mathbf{X}) - \delta_{ij})^2 + \lambda \sum_{j=1}^{m} \left(\sum_{i=1}^{N} x_{ij} \right)^2, \tag{4.1}$$

and use it instead of $s(\mathbf{X})$ hereinafter. For $0 < \lambda < \infty$ such a modification does not change the solution of the original problem, but restricts the center of mass of the resulting set of points to be at the origin. Since the second term is quadratic, the function $\hat{s}(\mathbf{X}) - \mathrm{trace}(\mathbf{X}^{\mathrm{T}}\mathbf{T})$ is bounded. This modification results in an increment by 2λ of every element of the diagonal blocks, $\mathbf{H}^{kk}$.

4.2. Coarsening strategy. We use a hierarchy of grids $\Omega_1 \supset ... \supset \Omega_R$, constructed in a multiresolution manner, where R is the coarsest resolution level, and denote by N_r the number of grid points at the r-th level. The transfer from resolution level r to the coarser level $r+1$ or the finer level $r-1$ is performed using an $N_{r+1} \times N_r$ matrix $\mathbf{P}_r^{r+1}$ (referred to as *restriction* operator in the multigrid literature) and an $N_{r-1} \times N_r$ matrix $\mathbf{P}_r^{r-1}$ (*interpolation* operator), respectively. These matrices are sparse (typically, every row contains from 1 up to 4 non-zero elements) and are often chosen to satisfy

$$\mathbf{P}_r^{r+1} = (\mathbf{P}_{r+1}^r)^{\mathrm{T}}. \tag{4.2}$$

The optimization problem is transferred to the next coarser level by applying a restriction operator $\tilde{\mathbf{P}}_r^{r+1}$ (not necessarily equal to $\mathbf{P}_r^{r+1}$) to the matrices $\boldsymbol{\Delta}$ and $\mathbf{W}$. Consequently, we have a hierarchy of problems of the form

$$s_r(\mathbf{X}_r, \mathbf{T}_r) \equiv \hat{s}(\mathbf{X}_r; \boldsymbol{\Delta}_r, \mathbf{W}_r) - \mathrm{trace}(\mathbf{X}_r^{\mathrm{T}}\mathbf{T}_r), \tag{4.3}$$

that need to be solved at each level.

4.3. V-cycle. The simplest multigrid algorithm is the V-cycle. The complete nonlinear multigrid optimization algorithm can be defined recursively in the following manner:

`MG_Vcycle`$(\mathbf{X}_r, \mathbf{T}_r, \boldsymbol{\Delta}_r, \mathbf{W}_r, K_r, K'_r)$

- If $r = R$ (coarsest level) solve

$$\min_{\mathbf{X}_R} s_R(\mathbf{X}_R, \mathbf{T}_R)$$

  and return.
- Otherwise:
  - Relaxation: Apply K_r iterations of an unconstrained optimization algorithm to $s_r(\mathbf{X}_r, \mathbf{T}_r)$ initialized with $\mathbf{X}_r$ and obtain $\mathbf{X}'_r$.
  - Compute

$$\begin{aligned} \mathbf{G}'_r &= \nabla s_r(\mathbf{X}'_r); \\ \mathbf{X}'_{r+1} &= \mathbf{P}_r^{r+1}\mathbf{X}'_r; \\ \mathbf{G}'_{r+1} &= \nabla s_{r+1}(\mathbf{X}'_{r+1}); \\ \mathbf{T}_{r+1} &= \mathbf{G}_{r+1} - \mathbf{P}_r^{r+1}\mathbf{G}_r. \end{aligned}$$

  - Apply the multigrid method on the coarser level,

$$\mathbf{X}''_{r+1} \leftarrow \texttt{MG\_Vcycle}(\mathbf{X}'_{r+1}, \mathbf{T}_{r+1}, \boldsymbol{\Delta}_{r+1}, \mathbf{W}_{r+1}, K_{r+1}, K'_{r+1})$$

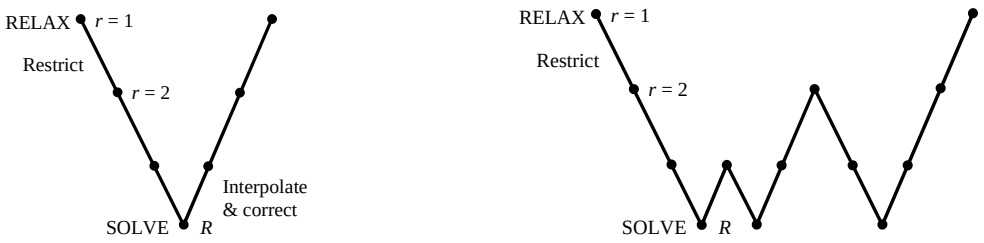
  - Correction:

$$\begin{aligned} \mathbf{E}_r &= \mathbf{P}_{r+1}^r(\mathbf{X}''_{r+1} - \mathbf{X}'_{r+1}); \\ \mathbf{X}''_r &\leftarrow \mathbf{X}'_r + \mathbf{E}_r. \end{aligned}$$

  - Relaxation: Apply K'_r iterations of an unconstrained optimization algorithm to $s_r(\mathbf{X}, \mathbf{T})$ initialized with $\mathbf{X}''_r$ and obtain $\mathbf{X}'''_r$.

The procedure `MG_Vcycle`$(\mathbf{X}, \mathbf{T}, \boldsymbol{\Delta}, \mathbf{W}, K, K')$ is initialized with some $\mathbf{X}$ and $\mathbf{T} = 0$ on the finest grid. The whole algorithm is repeated for a few iterations. When multigrid works well, a small number of V-cycles is required to obtain an accurate solution.

From the point of view of computation complexity it may be advantageous to use SMACOF iterations as the relaxation procedure at higher-resolution levels and Newton iterations at coarser resolution levels.



(a) V-cycle (b) F-cycle

FIG. 4.1. *Schematic representation of a V-cycle (a) and a F-cycle (b).*

4.4. F-cycle. Another popular MG cycle is the F-cycle [6]. In the recursive definition, the F-cycle procedure $\texttt{MG\_Fcycle}(\mathbf{X}_r, \mathbf{T}_r, \mathbf{\Delta}_r, \mathbf{W}_r, K_r, K'_r)$ is very similar to V-cycle, except the recursive call, which has the form

$$\mathbf{X}''_{r+1} \leftarrow \texttt{MG\_Fcycle}(\mathbf{X}'_{r+1}, \mathbf{T}_{r+1}, \mathbf{\Delta}_{r+1}, \mathbf{W}_{r+1}, K_{r+1}, K'_{r+1}),$$
$$\mathbf{X}''_{r+1} \leftarrow \texttt{MG\_Vcycle}(\mathbf{X}''_{r+1}, \mathbf{T}_{r+1}, \mathbf{\Delta}_{r+1}, \mathbf{W}_{r+1}, K_{r+1}, K'_{r+1}).$$

The difference between V-cycle and F-cycle is shown in Figure 4.1.

5. Results. The simulations in this paper focus on the face recognition application. We performed three experiments. The goal of the first two experiments was comparing the standard SMACOF algorithm with our multigrid implementation (V-cycle) on problems of different size. Random initialization (Experiment I) and initialization using the original point coordinates (Experiment II) were used. A facial surface patch was sampled several times by a different number of points ($N = 225, 625, 1425$ and 3249) for each test. The geodesic distances between all the points were computed using the Fast Marching algorithm [22]. The embedding space dimensionality was $m = 3$ in all the experiments.

Three resolution levels were used in the multigrid algorithm; each subsequent level containing $\frac{1}{4}$ of the points of the previous one. The grids were formed in a dyadic manner. The relaxation of each level was based on gradient descent SMACOF-like iterations described in Section 4. The optimization was terminated when the stress function at subsequent iterations decreased by less than 1% (this stopping criterion is common in MDS literature, see e.g. [3]). For comparison, a conventional SMACOF algorithm was executed with the same initialization and was terminated when reaching the same stress achieved by the multigrid algorithm.

The algorithm performance was evaluated by the stress value versus the execution time and the computational complexity in terms of multiplication operations (FLOPs). The square root operation was estimated as $C_s = 25$ FLOPs. All tests were performed on a PC with a 2 GHz Mobile Intel Pentium 4 processor and 1 GB RAM. The algorithms were implemented in MATLAB under Windows XP.

Tables 5.1 and 5.2 and Figure 5.1a-b show a comparison between a conventional SMACOF algorithm and the multigrid version in terms of convergence time and complexity in Experiments I and II, respectively. Figure 5.1c shows the boosting obtained by the multigrid implementation.

Several conclusions can be drawn from these results. First, multigrid implementation demonstrates significant performance improvement (over 6 times in Experiment II). The improvements contributed by the multigrid algorithm, when compared to standard SMACOF, become more pronounced as the size of the problem increases. That is, for large-scale problems the improvement is dramatic. Secondly, the multigrid algorithm converges approximately in the same number of iterations independent of N (in both experiments), while the number of standard SMACOF iterations grows with N. Thirdly, in Experiment I, the multigrid algorithm appears to be less sensitive to initialization (the standard deviation of the total convergence time divided by the mean convergence time is by an order of

TABLE 5.1
Experiment I: Comparison of multigrid and standard SMACOF algorithm on isometric embedding problems of different size, random initialization, results averaged over 10 runs. Overall execution time shown in seconds, overall complexity shown in MFLOPs.

N	SMACOF			MG		
	Iterations	Time	Complexity	Iterations	Time	Complexity
225	71.3	7.6 ± 1.78	113.26 ± 26.2	13.6	5.28 ± 0.36	65.83 ± 4.7
625	73	52.94 ± 10.16	894.53 ± 172.77	13.2	30.63 ± 1	480.86 ± 15.36
1425	111.5	421.76 ± 110.5	$(7.07 \pm 1.86) \times 10^3$	14.2	164.52 ± 5.53	$(2.67 \pm 0.079) \times 10^3$
3249	156	$(2.79 \pm 1.17) \times 10^3$	$(5.13 \pm 2.11) \times 10^4$	15.2	854.41 ± 29.67	$(1.49 \pm 0.04) \times 10^4$

magnitude smaller for the multigrid algorithm than for the standard SMACOF). Finally, the multigrid algorithm appears to be less likely to converge to local minima if random initialization is used; this is a similar phenomenon to what was observed when using multiresolution initialization [10].

In Experiment III, different variations of the multigrid algorithm were compared on a large problem ($N = 3249$). Specifically, we compared V-cycle with 3 and 4 resolution levels and F-cycle with 4 resolution levels. The results are shown in Table 5.3 and Figure 5.2. F-cycle with four resolution levels shows the best results; it outperforms the standard SMACOF algorithm by 8.12 times in sense of execution time and by 10.17 times in sense of overall complexity.

TABLE 5.2
Experiment II: Comparison of multigrid and standard SMACOF algorithm on isometric embedding problems of different size, initialization with the original points in $\mathbb{R}^3$. Overall execution time shown in seconds, overall complexity shown in MFLOPs.

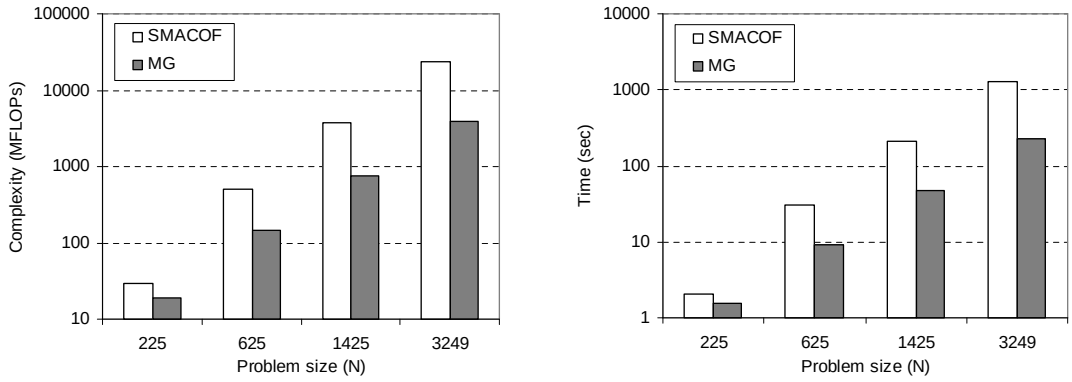
N	SMACOF			MG		
	Iterations	Time	Complexity	Iterations	Time	Complexity
225	18	2.03	29.62	4	1.53	19.22
625	41	30.41	507.03	4	9.26	145.72
1425	58	212.80	3.71×10^3	4	47.75	753.5
3249	72	1.29×10^3	2.38×10^4	4	224.37	3.90×10^3

TABLE 5.3
Experiment III: Comparison of different multigrid algorithms, initialization with the original points in $\mathbb{R}^3$. Overall execution time shown in seconds, overall complexity shown in MFLOPs.

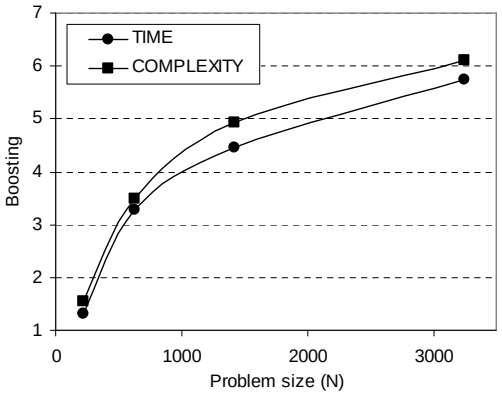
Algorithm	Iterations	Time	Complexity
SMACOF	93	1.68×10^3	3.03×10^4
V-cycle ($R = 3$)	4	276.15	3.90×10^3
V-cycle ($R = 4$)	4	220.42	3.89×10^3
Full MG ($R = 4$)	3	205.15	2.99×10^3

6. Conclusions. We propose a multigrid framework for efficient solution of multidimensional scaling problems. Tested on on large-scale embedding problems arising in 3D face recognition, our multigrid implementation demonstrates an order of magnitude better performance compared to the conventional SMACOF algorithm.

Though presented exclusively on isometric embedding problems, the proposed framework is applicable to general MDS problems. The range of applications in which our multigrid MDS algorithm can be used is very wide, and includes, to mention a few, problems in data visualization, machine learning, computational chemistry, etc. Among immediate applications currently under research is the general problem of isometry-invariant shape recognition and real-time texture mapping on complicated 3D objects.



(a) Complexity comparison (b) Time comparison



(c) Boosting

FIG. 5.1. *Experiment II: Comparison of multigrid and standard SMACOF algorithm on isometric embedding problems of different size, initialization with the original points in* $\mathbb{R}^3$. *(a) Complexity comparison, (b) execution time comparison, (c) boosting factor.*

REFERENCES

[1] D. K. Agrafiotis, D. N. Rassokhin, and V. S. Lobanov, *Multidimensional scaling and visualization of large molecular similarity tables*, Journal of Computational Chemistry **22** (2001), no. 5, 488–500.

[2] N. S. Bakhvalov, *On the convergence of a relaxation method under natural constraints on an elliptic operator*, USSR Comput. Math. and Math. Phys. **6** (1966), 861–883.

[3] I. Borg and P. Groenen, *Modern multidimensional scaling - theory and applications*, Springer-Verlag, Berlin Heidelberg New York, 1997.

[4] A. Brandt, *Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems*, Proc. 3rd Int'l Conf. Numerical Methods in Fluid Mechanics (Berlin), Springer Verlag, 1973, pp. 82–89.

[5] ———, *Multi-level adaptive solutions to boundary-value problem*, Math. Comp. **37** (1977), 333–390.

[6] ———, *Multigrid techniques: 1984 guide with applications to fluid dynamics*, GMD-Studie 85, Sankt Augustin, West Germany, 1984.

[7] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, *Expression-invariant 3D face recognition*, Proc. Audio and Video-based Biometric Person Authentication, 2003, pp. 62–69.

[8] ———, *Three-dimensional face recognition*, Tech. Report CIS-2004-04, Dept. of Computer Science, Technion, Israel, 2004.

[9] ———, *On isometric embedding of facial surfaces into* $\mathbb{S}^3$, Scale Space, 2005, to appear.

[10] M. M. Bronstein, *Three-dimensional face recognition*, Master's thesis, Department of Computer Science, Technion - Israel Institute of Technology, 2004.

[11] M. M. Bronstein, A. M. Bronstein, and R. Kimmel, *Expression-invariant representation of faces*, IEEE Trans. PAMI (2004), submitted.

[12] J. De Leeuw, *Recent developments in statistics*, ch. Applications of convex analysis to multidimensional scaling, pp. 133–145, North-Holland, Amsterdam, 1977.

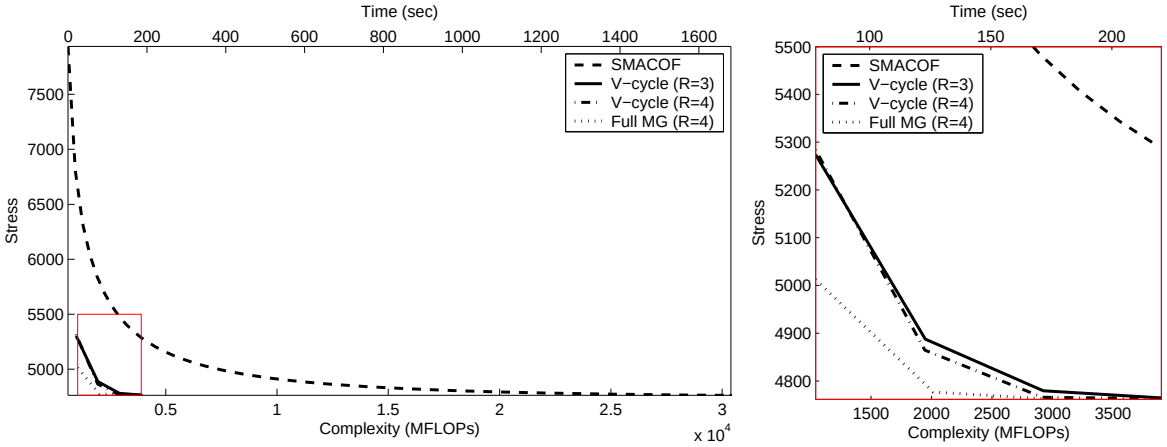


FIG. 5.2. *Experiment III: Comparison of different algorithms on problem with $N = 3249$ points, initialization with the original points in $\mathbb{R}^3$. Shown is the stress starting from the first iteration, as a function of computational complexity (bottom scale). The corresponding approximate CPU time is given on the upper scale. A zoom into a plot region marked with red is shown on the right.*

[13] J. De Leeuw and I. Stoop, *Upper bounds on Kruskal's stress*, Psychometrika **49** (1984), 391–402.

[14] A. Elad and R. Kimmel, *Bending invariant representations for surfaces*, Proc. CVPR, 2001, pp. 168–174.

[15] _____, *Geometric methods in bio-medical image processing*, vol. 2191, ch. Spherical flattening of the cortex surface, pp. 77–89, Springer-Verlag, Berlin Heidelberg New York, 2002.

[16] _____, *On bending invariant signatures for surfaces*, IEEE Trans. PAMI **25** (2003), no. 10, 1285–1295.

[17] R. P. Fedorenko, *On the speed of convergence of an iterative process*, USSR Comput. Math. and Math. Phys. **4** (1964), 559–564.

[18] P. Groenen and P. H. Franses, *Visualizing time-varying correlations across stock markets*, Journal of Empirical Finance **7** (2000), 155–172.

[19] R. Grossman, N. Kiryati, and R. Kimmel, *Computational surface flattening: a voxel-based approach*, IEEE Trans. PAMI **24** (2002), no. 4, 433–441.

[20] L. Guttman, *A general nonmetric technique for finding the smallest coordinate space for a configuration of points*, Psychometrika (1968), 469–506.

[21] A. Kearsley, R. Tapia, and M. Trosset, *The solution of the metric STRESS and SSTRESS problems in multidimensional scaling using Newton's method*, Computational Statistics **13** (1998), no. 3, 369–396.

[22] R. Kimmel and J. A. Sethian, *Computing geodesic on manifolds*, Proc. US National Academy of Science, vol. 95, 1998, pp. 8431–8435.

[23] H. Lahdesmaki, O. Yli-Harja, I. Shmulevich, and W. Zhang, *Distinguishing key biological pathways by knowledge-based multidimensional scaling analysis: application to discriminate between primary breast cancers and their lymph node metastases*, Proc. Bioinformatics, 2003.

[24] N. Linial, E. London, and Y. Rabinovich, *The geometry of graphs and some its algorithmic applications*, Combinatorica **15** (1995), 333–344.

[25] S. Nash, *A multigrid approach to discretized optimization problems*, Journal of Optimization Methods and Software **14** (2000), 99–116.

[26] E. L. Schwartz, A. Shaw, and E. Wolfson, *A numerical solution to the generalized mapmaker's problem: flattening nonconvex polyhedral surfaces*, IEEE Trans. PAMI **11** (1989), 1005–1008.

[27] J. Walter and H. Ritter, *On interactive visualization of high-dimensional data using the hyperbolic plane*, Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2002, pp. 123–131.

[28] G. Zigelman, R. Kimmel, and N. Kiryati, *Texture mapping using surface flattening via multi-dimensional scaling*, IEEE Trans. Visualization and computer graphics **9** (2002), no. 2, 198–207.