A Morphable Part Model for Shape Manipulation

Alexander Berner, Oliver Burghard,
Michael Wand, Niloy J. Mitra,
Reinhard Klein, Hans-Peter Seidel

MPI–I–2011–4-005 November 2011

Authors' Addresses

Alexander Berner
Universität Bonn
Bonn, Germany

Oliver Burghard
Universität Bonn
Bonn, Germany

Michael Wand
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Niloy J. Mitra
University College London
London, United Kingdom

Reinhard Klein
Universität Bonn
Bonn, Germany

Hans-Peter Seidel
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Abstract

We introduce morphable part models for smart shape manipulation using an assembly of deformable parts with appropriate boundary conditions. In an analysis phase, we characterize the continuous allowable variations both for the individual parts and their interconnections using Gaussian shape models with low rank covariance. The discrete aspect of how parts can be assembled is captured using a shape grammar. The parts and their interconnection rules are learned semi-automatically from symmetries within a single object or from semantically corresponding parts across a larger set of example models. The learned discrete and continuous structure is encoded as a graph. In the interaction phase, we obtain an interactive yet intuitive shape deformation framework producing realistic deformations on classes of objects that are difficult to edit using existing structure-aware deformation techniques. Unlike previous techniques, our method uses self-similarities from a single model as training input and allows the user to reassemble the identified parts in new configurations, thus exploiting both the discrete and continuous learned variations while ensuring appropriate boundary conditions across part boundaries.

Keywords

inverse procedural modeling, shape analysis, subspace symmetries

Contents

1 Introduction **2**

2 Related Work **5**

3 Morphable Part Model **7**

4 Learning a Model **14**

5 Applications **18**

6 Implementation and Results **20**

7 Conclusions and Future Work **23**

1 Introduction

Developing simple yet expressive deformation models to facilitate interactive and intuitive manipulation of geometry remains one of the important research areas of geometric modeling, interaction, and animation. Typically, the user specifies desired positions for a few handle points on the input pose, and the goal is to automatically deform the rest of the input model into a plausible new pose. Although a large variety of deformation techniques exists, the methods primarily differ based on what geometric properties are preserved during deformation. Notable recent approaches include preserving local surface details [29], keeping local elements as-rigid-as-possible [20, 28], achieving isometric deformations [22, 34], or respecting global relations across object parts [16, 37]. Unfortunately, many shapes, especially organic ones, show up in significant shape and pose variations. Many such classes of shapes cannot be related by local-rigidity, isometry, or similar distance measures based on just preserving local differential properties. Instead, the range of possible deformations are what actually characterizes the respective objects, and cannot be specified a priori. This motivates a data driven approach to first *learn* the space of allowable deformations, and subsequently *restrict* deformations only to the learned deformation space.

Given a set of reference model poses, one possibility is to establish a global correspondence across the multiple poses, and then build a statistical model to capture the dominant variations using a set of extracted parameters. Such a model is commonly referred to as a *morphable model*. The most commonly used statistical model is linear principal component analysis (PCA), which computes a global Gaussian model of the vertex positions of the mesh. In various applications, morphable models have produced impressive results [7, 13, 18]. However, there are some limitations: A global Gaussian model directly captures the correlations of all model points. This leads to a combinatorial blow-up for objects with many independent degrees of freedom: For example, in a human body shape, a large number of combinations of poses of the individual body parts need to be observed in the

training data so that a correct model can be learned. Furthermore, it is not possible to recombine parts because the analysis is based on global correspondences between all of the described shapes. For example, in Figure 5.1c deformations of the spider's body and legs are better described by separate morphable models. Nevertheless, the parts are mutually correlated, e.g., the spider's body cannot shrink without adjusting the size of the legs.

In this work, we propose a part-based morphable model to overcome these limitations. We capture variations of individual parts of an object along with the mutual dependencies across the parts, where both the part-variations and the inter-part dependencies are learned from training data (see supplementary video). We encode the variations as a graph where each node represents an object part, while the edges store the relations among the parts. Thus the continuous variability is captured by attributes at the nodes and the edges, while the discrete variability is stored as the graph connectivity. We semi-automatically learn such a *morphable part model* starting from a input set of training poses. The parts and their relations are combined using an elastic deformation model that connects different parts while ensuring seamless stitches and globally distributing deviations from the observation. Our model corresponds to a Gaussian Markov random field (MRF) rather than a global Gaussian model, which gives us the opportunity to describe part behavior and part interaction locally, and gives us a well defined interface to rearrange parts in different combinations. We reduce the resultant system to solving a large sparse Laplacian matrix and achieve interactive performance using a Schur complement decomposition.

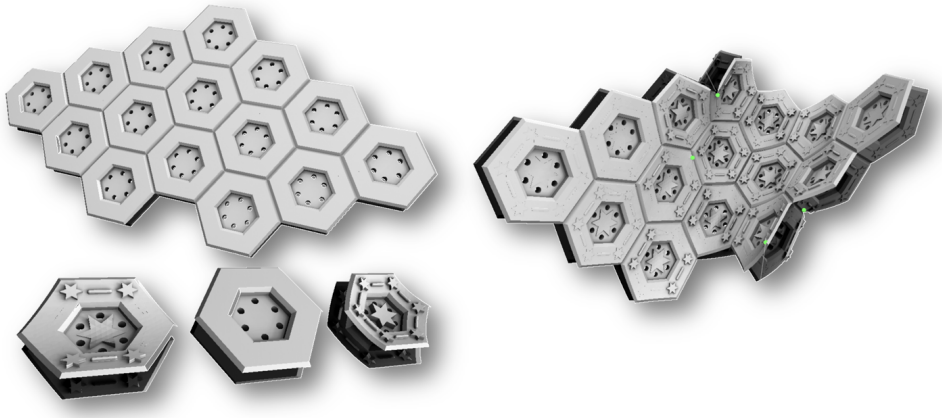


Figure 1.1: *An assembly of artificial parts to illustrate our method: Three variants (lower left) of a part with different bending are learned from a single example model. In an assembly, parts react to neighboring tiles and morph to best fit the learned model.*

We use our framework towards an intuitive deformation framework by restricting deformations to the learned morphable part space obtained by examining partial, deformable symmetries of the input shape (e.g., using subspace symmetries [6]) or corresponding parts from a larger collection of example shapes. As a special case, partial symmetrization is achieved when corresponding parts are approximated by the average part from the respective shape space while maintaining the global structure of the object. In another application, we support both discrete and continuous shape edits. Based on user annotations of a shape database, we first extract continuous and discrete rules to encode the space of deformations prescribed by the input poses. We then construct new graph variants that are compatible with the learned model and subsequently embed the graph in 3D using a least-squares formulation. We test our framework on a variety of examples enabling interactive, intuitive, and expressive shape manipulations (see supplementary video).

2 Related Work

Over the last decades, a vast range of research efforts have focused on facilitating shape manipulations under different deformation models (see [10] for a survey). We review only a small selection, focusing on approaches directly related to our goal.

Deformation models: In a highly influential work, Sederberg and Parry [27] use trivariate Bernstein polynomials for free-form deformation of the embedding volume of an object and thus warp the immersed object. Subsequently improvements have been proposed using richer deformation bases to obtain plausible interpolation behavior of the embedding space ([25, 5] and references therein). In an alternate approach, researchers model surface or enclosed volume deformations with approximate elastic behavior using variational formulations, e.g., [20, 9]. For surface deformation, popular approaches locally preserve surface details using a Laplacian formulation [29], or allow deformations that keep local elements as-rigid-as-possible [28]. Our algorithm can use any such methods for local deformation. In contrast to other alternatives, we only allow deformations restricted to the learned space of morphable models and conform to the inter-part dependencies using a global coupling.

Statistical shape models: Morphable models, where deformation models are constructed using a statistical characterization of the input set of model poses, have been used extensively in computer graphics and computer vision ([13, 18] and references therein). Based on available correspondence across the various input poses, the methods perform global dimensionality reduction to compactly encode dominant shape variations. For example, mesh-based inverse kinematics by Sumner et al. [30] uses a global PCA model to guide deformation modeling. Our method generalizes this ideas to multiple coupled pieces. Modal analysis has also been used to speed up the simulation of deformable objects [4]. Feng et al. [15] use kernel canonical correlation analysis to control animations with a small number of handles, while Baran et al. [3] present a animation modeling

technique by building local deformation models for patches to transfer animations across classes. All of these methods assume global correspondence information across inputs. Zhang et al. [36] propose FaceIK to combine example faces by spatial weighting and similarity-based weighting and explicitly avoids addressing how to combine local PCA models, which is the focus of this work. Recently, Tena et al. [31] introduce a data-driven approach to learn a piecewise PCA face model from facial motion capture data to enable local control for facial expression generation. Our model differs as follows: (i) We model rules for connecting parts such that we can describe a large class of shapes with different arrangements of parts; (ii) Our model is based on an elastic deformation model that puts parts together seamlessly, without need for interpolation and with explicitly modeled interaction between cliques of parts.

Structure-aware deformation: A number of *structure-aware* deformation models exist that try to *understand* the structure of the input geometry in order to deform the input in a smart way, e.g., Kraevoy et al. [24] look at differential surface properties to protect vulnerable parts against unnatural bending, while Xu et al. [35] infer joint properties using a slippage analysis. The iWires system [16] uses *wires* or feature curves to learn and maintain intra- and inter-wire relations extracted from man-made objects in order to enable natural deformations. Improvements have been proposed using symmetry hierarchies [33] or component based deformations [37]. In contrast, we exploit correspondence between potentially strongly deformed parts, rather than parts related only by rigid transforms. We are also motivated by recent efforts in inverse procedural modeling [8] where shapes are decomposed into rigid building blocks and a shape grammar that describes how to create similar objects. Again the technique is limited to handle strict rigid mappings. We overcome this restriction by learning a correspondence from user input and realizing an embedding of the deforming parts as concrete geometry into three space.

3 Morphable Part Model

Our morphable part-based model (MPM) consists of two components: (i) a set of *discrete* rules describing how a complete object can be assembled out of *parts* connected through *docking sites*, (ii) a set of subspaces of observed part variations in which the respective individual parts (as well as the connecting geometry) can morph. The model statistically captures both inter- and intra-part variations and allows us to compute optimal embedding of part graphs for a given logical assembly of parts and additional user constraints. We now introduce the components in details, while in Section 4 we describe how to semi-automatically learn such a MPM.

i) Discrete model: In this work, *part* refers to a surface patch where topological connectivity is maintained but the geometric positions can vary, e.g., a triangle mesh with fixed connectivity but varying vertex positions. Within a part, multiple regions can be designated as *docking sites*, e.g., boundary edges of the mesh (see Figure 3.1a,b). Docking sites act as connectors to assemble multiple parts to build a composite model: Each docking site is tagged with the type of part it can connect to and designated corresponding docking site(s) on the target. Overall, this encodes a shape grammar, which is not necessarily context free [8]. It encodes all pairs of parts that can be combined through a pair of matching docking sites: Each pair of connecting docking sites forms one connection opportunity. All the combinations that have been observed and only these will be added to the grammar, which can be compactly encoded as a graph of docking sites (see Figure 3.1c).

ii) Continuous model: The continuous component captures both the *a) single*, i.e., the intra-part variations for each individual type of part and the *b) part pairs*, i.e., the inter-part variations across pairs of part types. The overall model is obtained by multiplying the two likelihoods, which yields a Gaussian Markov Random field (MRF). In our experiments, we found it is desirable to build the statistical models in the gradient domain (e.g., using the Laplacian [29]) rather than in the spatial domain. This results in a global diffusion of errors from imperfectly

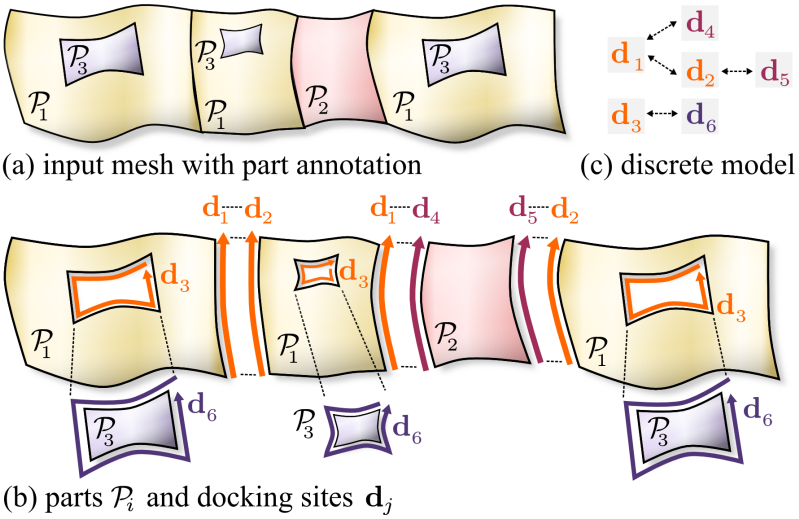


Figure 3.1: *Decomposition of an input mesh (a) into parts and docking sites (b). We obtain a shape grammar (c) that encodes all permissible pairwise connections between parts. Each part and each region of pairwise connection is described by a morphable model.*

matched parts, thus adding more flexibility to the modeling framework . We now elaborate on the single and pairwise variations.

a) Single-part variations: For each type of part, we build a morphable model from a collection of instances $\{\mathcal{P}_1, \ldots, \mathcal{P}_{n_p}\}$, each representing a surface patch. We first align the patches, establish a dense correspondence across them, and then transfer mesh connectivity from the first instance to the others (see [6] for details). We encode each part instance $\mathcal{P}$ as a long vector $\mathbf{V} = (\mathbf{v}_1, ..., \mathbf{v}_{n_v}) \in \mathbb{R}^{3n_v}$ representing its geometric positions and a consistent mesh topology. Later, we use the topology to consistently transfer the part instances to the gradient domain. Thus, in our model, each part type has a fixed mesh topology and parameters encoding the shape variation (a Gaussian density on vertex set $\mathbf{V}$), as described next. To simplify notations, we use $\mathbf{TV}$ to denote the modified vertices of a part instance due to transformation $\mathbf{T}$.

Variations in each type of part are mainly due deformations characteristic to the type and rigid placements of the parts. We capture the characteristic variations using a set of linear shape parameters $\Lambda := \{\lambda_1, \ldots, \lambda_d\}$ in a learned d-dimensional linear shape space. If $\mathbf{T} \in SE(3)$ denotes a rigid placement (i.e., drawn from the group of rotations and translations of $\mathbb{R}^3$), the resultant embedding take the form,

$$\mathbf{V} = \mathbf{T}\left(\boldsymbol{\mu} + \sum_{k=1}^{d} \lambda_k \mathbf{b}_k\right) \tag{3.1}$$

$\boldsymbol{\mu}$ is the mean shape of part $\mathcal{P}$ and $\{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ is a set of orthogonal vectors in the shape space $\mathbb{R}^{3n_v}$ that describe principle modes according to which the geometry of the shape can vary (see Section 4).

Such a subspace model captures the *ideal* geometry according to the learned model. However, due to explicit user constraints or implicit constraints of assembling multiple such types of parts together, often such constraints can only be approximately satisfied. Hence, we use a variational least-squares formulation:

$$E_{sub}(\Lambda, \mathbf{T}) = \sigma_{sub}^{-2} \left(\mathbf{V} - \mathbf{T}\boldsymbol{\mu} - \sum_{k=1}^{d} \lambda_k \mathbf{T}\mathbf{b}_k \right)^2 . \tag{3.2}$$

One can minimize E_{sub} over the free variables $\Lambda \in \mathbb{R}^d$ and $\mathbf{T} \in SE(3)$ to penalize deviations from the subspace. Further, we constrain the range in which the subspace coordinates Λ lie using the energy

$$E_{var}(\Lambda) = \sum_{k=1}^{d} \left(\frac{\lambda_k}{2\sigma_k} \right)^2 . \tag{3.3}$$

The final penalty function is the sum of E_{sub} and E_{var}. Statistically, E_{var} captures shape variations using a Gaussian model with mean $\boldsymbol{\mu}$ and standard deviations $\{\sigma_1, \ldots, \sigma_d\}$ along the principal axes, which we will later learn from example data. Further, in the orthogonal complement of the subspace, we have a uniform Gaussian shape model with standard deviation σ_{sub}, which depends on the part type. In practice, we use $\sigma_{sub} \ll \sigma_i$, i.e., leaving the subspace is more expensive than moving within the subspace. The subspace and its parameters $\{\sigma_1, \ldots, \sigma_d\}$ are learned using PCA, while σ_{sub} is a user parameter determining model flexibility outside the subspace (typically: $\sigma_{sub} \in [3, 10]$).

In our experiments, we found the above model to be too restrictive especially when stitching together multiple pieces. Therefore, we reformulate the problem as an elastic matching problem in the gradient domain: Since we have a consistent set of edges E across all the instances of a particular part type, we propose the following elastic subspace attraction energy, which relates edge vectors, i.e., local differences of positions, instead of absolute coordinates:

$$E_{rel}(\Lambda, \{\mathbf{T}\}) = \sigma_{sub}^{-2} \sum_{(i,j)\in E} \omega_{i,j} \Bigg((\mathbf{v}_i - \mathbf{v}_j) - \left(\frac{\mathbf{T}_i + \mathbf{T}_j}{2} \right) \left[(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) + \sum_{k=1}^{d} \lambda_k \left(\mathbf{b}_{ki} - \mathbf{b}_{kj} \right) \right] \Bigg)^2 \tag{3.4}$$

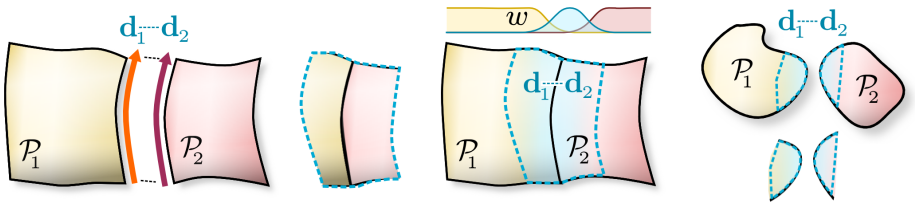


Figure 3.2: *Two types of docking sites: (a-b) docking at mesh boundaries, (c) this might include weights, (d) disconnected components can be docked by designating shared geometry.*

The energy E_{rel} is then used instead of the direct subspace energy E_{sub}. By measuring the errors locally instead of globally, low-frequency deformations are less penalized enabling elastic deformations [29]. The preservation of each difference vector is weighted by a cotangent weight $\omega(i, j)$, where the angles are measured in the configuration of the mean shape $\boldsymbol{\mu}$. For non-manifold vertices, we use uniform weights. Note that to permit local changes in part orientation, we introduce free rotation variables $\mathbf{T}_i \in SO(3)$ at each vertex of the mesh [28]. As an alternative, we can also use a single transformation variable per part. Experimenting with both variants, the per-part method lead to slightly better results and faster convergence in our example models. Therefore, it is used in all examples shown.

b) Pair of parts variations: We now describe how to create composite models by stitching together several single parts. Such an assembly involves a discrete aspect to decide which combinations of parts are desirable, and a continuous aspect to optimize the geometry at the part joints.

Discrete Assembly Model: The discrete model provides a set of rules that control how parts can be assembled. The idea is simple: To connect two parts, we need to specify where they connect, i.e., determine the mesh topology of the connection. Furthermore, we need to describe the geometry of the connection. Both aspects require that we have observed such a connection at least once. Docking sites designate the area in which the parts are connected. Therefore, every pair of connected docking sites found in the training data will create one connection rule, represented by a graph edge between the two docking sites (Figure 3.1c). The resulting graph determines the possible mesh topologies of the created objects. To control the geometry, we build a statistical model of how the shape in the vicinity of the docked docking sites behaves. It aggregates all examples observed for one such docking rule.

Continuous Assembly Model: The final ingredient is a continuous model of how

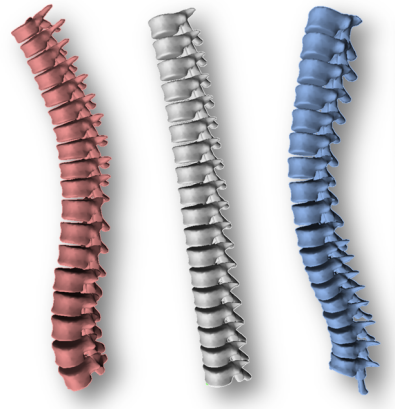
the part connections affect the overall shape. The continuous parameters of the parts naturally are coupled based on how the parts are assembled. We model each docking site as being governed by a (elastic) Gaussian shape model. We learn the model using a PCA analysis from example connections between parts.

We support two types of docking sites: We first discuss the most common type, the *boundary docking site*: In this case, two parts share a common boundary and thus the boundary vertices between two parts are constrained to be at the same position (Figure 3.2a), forming a watertight connection. In order to build a statistical model, we gather the geometry within a fixed distance to the boundary between the parts. We call this geometry the geometry associated with the docking site pair, or in short, *pair geometry* (Figure 3.2b). For each pair of docking sites we again build a morphable part model for its pair geometry, according to Equations 3.3 and 3.4. The pair model has the same form as the single-part model but it now applies to the region of geometry attaching two pairs. It is learned from all parts of example geometry where we observe the same type of connection (same pair of docking sites).

For better smoothness, we use smooth weights, i.e., the attraction to the part singleton models fades continuously to zero when approaching the boundaries. Contrarily, the attraction to the docking site model grows when moving towards the boundary of the parts, which is away from the boundaries of the geometry associated with the docking site (this is sketched in Figure 3.2c). We weight each vertex as $\exp(-d^2/\sigma^2_{bound})$, where d is the distance to the closest boundary point (of a part or pair geometry, respectively). We then multiply the edge weights $\omega_{i,j}$ with the average of the vertex weights in order to reduce the influence when approaching the boundary. The size σ_{bound} of the blending region is a user parameter; a good choice for seamless results is to blend within about one third of the tile diameter.

We now introduce the second type of docking site, which is a generalization of the first type. The second option is less stringent: We also allow pairs of parts that are disconnected in the input. We do so by forming pair geometry that includes parts of both models (Figure 3.2d). Algorithmically, we include all geometry in part 1 that is within at most a fixed distance from part 2 and vice versa. We now connect each such pair of points that connects from part 1 to 2 by a virtual, non-manifold edge and setup the Laplacian deformation model of Equation 3.4 to preserve these distance vectors, allowing a single global rotation for the whole part. This reduces the ability to deform the geometry a bit but ensures to preserve the empty space inbetween the disconnected parts. As an application example, the spine model in Figure 3.3 is a collection of disconnected meshes for each part so that we have to use this docking mechanism here for modeling.

Computing an Embedding: Having assembled together a set of parts into a mor-



(a) input, sym, anti-shape

(b) input

(c) symmetrized

(d) anti-sym: -0.5Λ

(e) caricature: 2Λ

Figure 3.3: *Symmetrization by setting mean subspace parameters. Caricatures are obtained by doubling the subspace coordinates. Using negative values yields an "anti-shape".*

phable part model, our remaining task is to compute a good embedding, i.e., determine geometric vertex positions for the prescribed topological connectivity. Note that vertices can belong to one or multiple parts (e.g., vertices on docking boundaries). In our formulation the individual parts, as well as the connecting docking sites, attract vertices to respective shape spaces. We denote the energy functions of the individual parts by $E_i, i \in \{1 \dots k\}$, the energies contributed by the docking sites by $E_{i,j}, (i,j) \in G$. Each such energy is the sum of terms according to Equations 3.3 and 3.4. G denotes the edges of the undirected graph of pairwise connections through docking sites.

$$E(M) = \sum_{i=1}^{k} E_i + \sum_{(i,j)\in G} E_{i,j}. \tag{3.5}$$

The unknowns of this system are: (i) the vertices $\mathbf{V}$ of the joint mesh, (ii) the respective rotation matrices, and (iii) the shape space parameters Λ_i and Λ_{ij}, which are global to each part and docking site, respectively.

Solving the system: In order to minimize the above energy, we alternate between solving for the vertex positions and shape parameters (i,iii) while keeping the transformations (ii) fixed, and the other way round. Specifically, starting from an initial guess of vertex positions and shape parameters, we solve for the optimal rigid transformations by shape matching (i.e., we fit rotation matrices to the local 1-rings in the meshes). Subsequently, we fix the transformations to solve for lambda and vertex positions by minimizing Eq. 3.5. This is a quadratic energy,

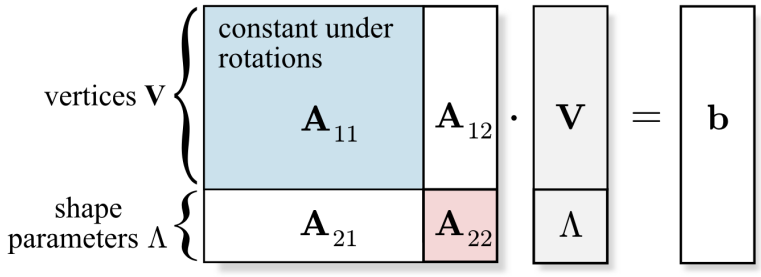


Figure 3.4: *Matrix decomposition. The system matrix consists of a sparse and constant Laplacian matrix* $\mathbf{A}_{11}$ *that remains constant and thus can be pre-factored. The matrices* $\mathbf{A}_{12} = \mathbf{A}_{21}^T$ *depend on the rotation variables and have to be updated dynamically. A Schur complement decomposition reduces the problem to solving systems with the sparsely pre-factored* $\mathbf{A}_{11}$*, solving a small linear system* $\mathbf{A}_{22}$*, and matrix-vector products with changing matrices* $\mathbf{A}_{12}$*.*

summing up terms in the form of Eq. 3.4 for each part and each docking site. We analytically compute the derivative of this energy and obtain a linear system we need to solve.

In the direct form, such an alternate optimization is slow and unsuited for interactive manipulation. Further, as explained below, we cannot globally prefactorize the system matrix (unlike [28]) even when the transformation are kept fixed. A non-linear joint optimization [32] is possible, but would add significant complexity. The key observation for an efficient solution is that the shape parameters are low-dimensional in comparison to the vertex positions. The system matrix of the linear system can be understood as a covariance matrix of the Gaussian distribution in vertex and shape parameter space (Figure 3.4): We can order the columns and rows such that the (large) upper left block $\mathbf{A}_{11}$ forms the vertex-to-vertex covariance, while the lower right block $\mathbf{A}_{22}$ correlates shape parameters, and the diagonal blocks $\mathbf{A}_{12} = \mathbf{A}_{21}^T$ encode correlations between shape parameters and vertex positions. The matrix $\mathbf{A}_{11}$ is constant under changing transformations $\mathbf{T}_i$. The other three have to be recomputed, because the shape basis vectors $\mathbf{b}_k$ are co-rotated by the $\mathbf{T}_i$ in Eq. 3.4, thereby changing matrix entries associated with the shape parameters. We therefore split the solution using a Schur complement decomposition (see [11], pp. 672ff). We solve the large, constant system (matrix $\mathbf{A}_{11}$) efficiently using a precomputed sparse Cholesky decomposition and solve the small problem using conjugated gradients. This approach typically leads to a substantial speedup by an order of magnitude in comparison to a solution without prefactorization.

4 Learning a Model

In this section, we present the necessary pre-processing steps to convert training data to a morphable part model, which is then used for manipulation (see Section 3). We allow the user to indicate part-level correspondence for sematic relations between geometrically dissimilar parts. Subsequently, we refine and establish a dense correspondence.

Learning Gaussian shape models for parts in dense correspondence is straightforward: we simple apply principal component analysis (PCA) to a set of training models in correspondence. The main challenge is establishing such a dense correspondence among parts with very different geometry. Most relevant approach is by Berner et al. [6] who automatically find symmetric parts that lie within a small subspace. The technique, however, can only handle moderate geometry variations where a feature matching heuristic suffices to identify matching candidates. Semi-automatic approaches [21] based on machine learning from user examples are possible, but the methods do not provide dense correspondences. While we expect future methods to address the issue of semantic correspondence, we allow manual annotations to produce necessary input.

Part correspondences: We first segment an input mesh into components, and then establish dense correspondence across the related components. While some inputs come pre-segmented into meaningful connected components, in other cases we expect the user to manually prescribe cutting lines. The user indicates a few ordered points, which we connect using (shortest) geodesic paths (using a Dijkstra algorithm). For distance measure, we use a combination of surface distance and feature based similarity score (see isophotic semi-metric [26]) to encourage curves to snap to ridge/valley curves. Thus, for each part type we have a set of mesh segments $\{S_1, S_2, \dots\}$.

For each part type, in order to establish a dense correspondence across its segments, we establish dense correspondence from a segment, say S_1, to all the oth-

ers, say S_i, and transfer the S_1-mesh structure to S_i. We start with a set $\Phi := \{S_1\}$. We then select the segment $S_i \notin \Phi$ that deviates *most* from all the members in Φ (based on bounding box diagonal length). In a bootstrapping stage, we use user-marked landmark correspondences between S_1, S_i to align $S_1 \rightarrow S_i$ using the indicated correspondences as constraints and refine the solution using deformable iterated-closest-point (ICP) method: We first use coarsely discretized thin-plate-splines as regularizer [12] to obtain a coarse alignment. Starting from this as new initial guess, we afterwards use a standard elastic (Laplacian) deformation model to capture the high-frequency details at mesh resolution. Using this dense correspondence between S_1, S_i, we transfer the mesh structure from $S_1 \rightarrow S_i$ and update $\Phi \leftarrow \Phi \cup S_i$.

Once we have a few elements in Φ (2-4 in our experiments), we build its morphable model $\Phi^\star$ using PCA since all the constituent meshes share the same structure at this stage. For any subsequent element $S_i \notin \Phi$, we find the element $\hat{S}$ in the subspace $\Phi^\star$ closest to S_i by using our new deformable model instead of the thin-plate splines during deformable ICP. Subsequently, we establish a dense correspondence between $\hat{S}, S_i$ — however, the two elements being similar at this stage, automatic deformable iterated-closest-point (ICP) method suffice. If the difference between $\hat{S}, S_i$ is large, the user can initiate landmark point based alignment, but this is rarely necessary.

We learn the PCA model by first applying rigid shape matching to each S_i with respect to S_1 and then use the resulting vertex positions of all S_i as input to a conventional principal component analysis [7, 2]. We iterate these two steps until convergence.

Refining correspondences: Slippable surfaces [17], i.e., tangential ambiguity along the target surface leading to different drift in each pairwise correspondence $S_1 \leftrightarrow S_i$, can distort the morphable model. Researchers in computer vision have investigated measures for *good* Gaussian shape spaces [19, 23, 14]. One way to characterize compactness of a Gaussian model uses its entropy, which is related to the determinant of the covariance matrix [23]. We adopt a simpler approach to minimize the sum of the variances in the model [19]. This regularizer removes unnecessary variance that is not justified by data matching or user constraints, as in our problem setting, as described next.

The aim of the optimization is to refine the computed pairwise correspondences by removing unnecessary variance, e.g., due to tangential drift. Let $f_i : S_1 \rightarrow S_i$ describes the pairwise correspondences between the base part segment S_1 and any other part segment S_i. While keeping the corresponding boundaries of the shapes,

we optimize the following energy function:

$$E_{pca} = E_{boundary} + E_{sample} + \mathrm{tr}\,\Sigma, \tag{4.1}$$

where tr denotes matrix trace. The three energies depend on the choice of correspondences $(f_2, f_3, \dots)$. The $E_{boundary}$ penalizes deviations from the user-annotated landmark correspondences, and also encourages boundary vertices to remain in the vicinity of boundary curves. The E_{sample} term penalizes uneven sampling using a Laplacian term — for each vertex, we compare its position to the average of the vertices in its 1-ring neighborhood and quadratically penalize the difference. The last term is the most important one as it penalizes the trace of the covariance matrix obtained from the correspondences [19] and helps remove tangential drift. Specifically, assuming that S_1 has vertices $\{x_1, \dots x_m\}$, which are mapped by the f_i to the corresponding pieces, we obtain:

$$\mathrm{tr}\,\Sigma := \sum_{i=1}^{n}\sum_{j=1}^{m}(f_i(x_j) - \mu_j) * (f_i(x_j) - \mu_j), \tag{4.2}$$

where μ is the mean shape obtained from the correspondences. Thus, the trace-energy penalizes deviation from the mean shape, thereby globally reducing the variance. This reduces parametrization drifts, even for examples with very different forms. While alternate approaches are possible, we leave a careful comparison to future studies.

In practice, we perform the optimization over the tangent space of $S_2, S_3, \dots$ using a gradient descent on E_{pca} and allowing vertices to freely move in $\mathbb{R}^3$. We, however, restrict each vertex to only move by a small margin dictated by the (local) sampling density of the mesh vertices. After each step, we project the vertices back to the mesh surface, and iterate. This removes tangential drift, and produces compact Gaussian shape models (see accompanying video).

Learning docking rules: After establishing dense correspondences between all part-pairs for each part type, we learn a shape grammar, i.e., a set of discrete rules as how the parts can be attached to each other. Since we already have a graph decomposition of the input model (see Figure 3.1), we simply search and collect patterns of how each node in the graph is connected to its 1-ring neighbors and store them as a rule set. Subsequently, we compute the shape parameters for each example part.

Between two docking parts P_i and P_j, we identify the docking area based on an "influence region" parameter r — we include all points of part P_i that are within distance of any point in P_j, or vice-versa (see Figure 3.1). Additionally, when a part is disconnected (e.g., a polygon soup), we introduce artificial links to connect

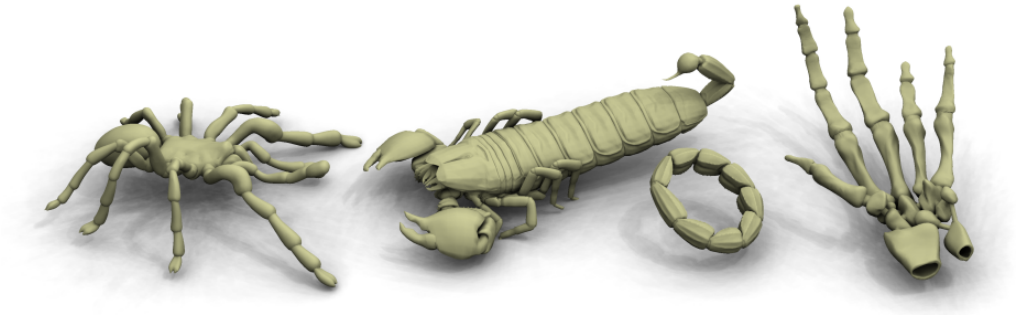


Figure 4.1: *Changing the discrete graph structure, assembling parts in a different configuration and solving for optimal embedding. Left: adding/removing segments to the spider legs; middle: stretching the scorpion body and removing parts from the tail; below: creating a cyclically connected tail; right: inserting a segment to the pointing finger and removing parts of the remaining fingers.*

each triangle/point to a central element of the part (e.g., the element closest to the centroid of the part). By preserving the distance vectors to the centroid, the actual "docked" parts cannot drift away. The variations in relative pose are automatically learned from the different example configurations, as described earlier.

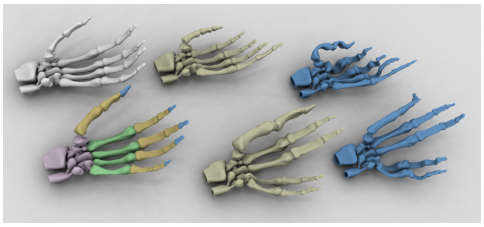
Seamless stitching: We need to ensure that two parts $\mathcal{P}_1, \mathcal{P}_2$ that share a common boundary fit together seamlessly. We do so by sharing the vertices along the boundary curve, i.e., using only a single unknown variable for each shared vertex. As we have dense correspondences along the two boundary curves, we simple take the union of the vertices and split the triangles accordingly. The statistical model for newly inserted vertices from $\mathcal{P}_2$ is obtained by interpolation of the two closest vertices from $\mathcal{P}_1$, and vice versa. Stitching is done after fixing a layout of connected patches, not a priori per patch. This offers greater flexibility, even allowing docking sites that connect to parts of themselves recursively.

5 Applications

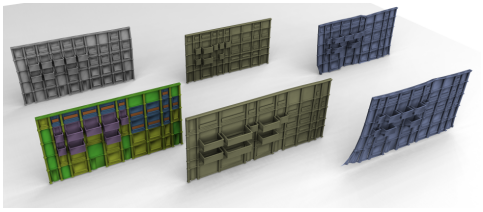
Free-Form deformation: In our variational formulation for embedding graphs of parts, it is easy to enable free-form shape deformation using additional energies to model handle constraints. In our implementation, we use simple least-squares position constraints of the form $\sum_i (f(\mathbf{x}_i) - \mathbf{y}_i)^2$, where pairs $\mathbf{x}_i \rightarrow \mathbf{y}_i$ are user prescribed handles, $\mathbf{x}_i$ the point on the original undeformed model, and $f(\mathbf{x}_i)$ denotes the computed embedding. Adding further constraints that prescribe rotations or transfinite constraints (such as area and curve constraints) is also conceivable. Note an important detail: If we use the part-based model as is, the original state of the model is not the optimum of the energy function but the model will be biased to deform towards the mean in each part and each docking site. This is not desirable for a free-form deformation tool, where the rest state should always be the original model. We therefore exchange the means: For each actual part instance, we determine the shape parameters λ within the learned subspace and replace the learned average mean with the actual part parameters, thus removing the bias. See accompanying video for typical behavior.

Partial symmetrization: In this application, we do the opposite: We use the average mean for all part classes and replace the original covariance matrices with isotropic Gaussian distributions with small variance. This encourages the individual parts to assume the same mean shape in each part. By changing the mean within the available subspace parameters, we can control the shape that all of the parts are trying to assume (see Figure 3.3). The opposite effect can also be created, by increasing the distance from the per-part mean, which creates caricatures of the input model (as done by Blanz et al. [7] for single component models).

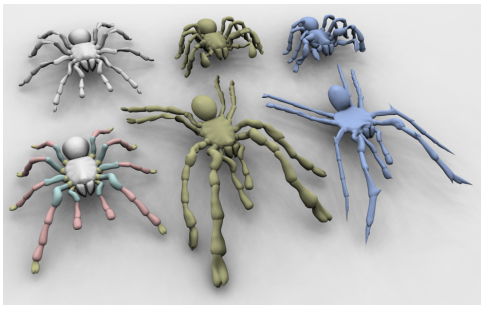
Inverse procedural modeling: By training a part-based morphable model, we construct not only a deformation model but also a set of discrete rules for assembling the parts. The continuous deformation model serves as a tool to compute an actual embedding of the abstract graph, while the residual energy of the optimal embedding indicates the overall distortion necessary to realize the graph.



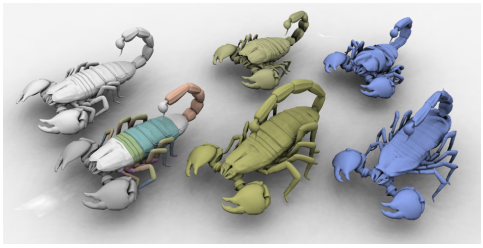
(a) hand



(b) facade



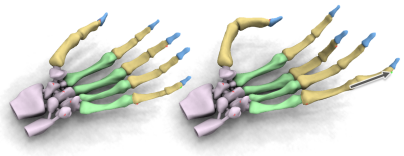
(c) tarantula



(d) scorpion

Figure 5.1: *Deformation examples. Lower left (colored parts): original input. Upper left (white): symmetrized. Middle column (yellow): deformed with our method. Right column (blue): standard elastic (Laplacian/ARAP) surface deformation.*

(a) plant model, locally symmetrized



(b) hand model, symmetrized editing

Figure 5.2: *Additional constraints placed on subspace coordinates: (a) enforcing similarity of local subspace coordinates, (b) editing with global symmetrization of the yellow parts.*

As described, our approach automatically learns a grammar according to which parts can be attached. We allow the user to manually specify a new graph based on the constructed grammar. Subsequently, we run the continuous optimization to compute its optimal embedding. In future it will be interesting to allow automatic spawning of such graphs based on statistics of model databases.

6 Implementation and Results

We have implemented our method single-threaded in C++, but using the multi-threaded Intel Math Kernel Library for solving linear systems. Table 6.1 lists performance statistics on a workstation with an Intel quad core i7-2600K (3.4GHz) and 8GB of RAM. Results are shown in Figures 1.1, 3.3, 4.1-5.2. Please see the accompanying video — the deformation behavior is best experienced in an interactive scenario.

We used publicly available models: The *hand* model is a manifold meshes with boundary docking sites, while the *plant*, the *tarantula* and the *spine* models consist of separate parts that are connected by generalized docking sites. The *scorpion* model uses both techniques - boundary docking sites in the body region and separate parts in legs and tail. In addition, we synthesized the *facade* and *hexa-grid* to complement the evaluation set. For the models from external data sources, we estimate correspondence as described in Section 4, while for the synthesized models we use the known correspondence information.

Figure 5.1 shows deformation results obtained with our technique. The model in the lower left corner is the original input, with corresponding parts tinted in

model	**vertices**	**parts**	**def. time(ms)**	**type**
hand	23081	20	150	bnd
facade	45493	128	300	cmp
tarantula	28996	57	170	cmp
scorpion	66453	59	350	bnd & cmp
spine	16254	18	150	cmp
hexa-grid	51465	15	350	bnd
plant	22330	29	180	cmp

Table 6.1: *Model statistics. Timings: average for one iteration. Type: boundary docking sites (bnd), disconnected components with generalized docking sites (cmp), or both.*

matching colors. The yellow variations are editing results from sparse constraints (see supplementary video) and the white model shows a symmetrized variant. We use as-rigid-as-possible Laplacian surface editing model [28] as representative for elastic free-form deformation techniques. The according results are shown in blue. In comparison MPM results, we observe the artifacts: (i) When models are squeezed, unlike our data-driven approach, elastic deformation produces unfavorable results with surfaces folding and wavy artifacts (e.g., hand and tarantula examples). (ii) When models are stretched, the results, in absence of folding, are still not plausible in comparison to the learning-based results. Note that with sparse set of constraints, MPM can produce spiking artifacts, while the elastic model resists such forces (unlike the MPM, which adapts the shape within the learned subspace). Such artifacts could be reduced by more elaborate constraints (e.g., area handles). Note that unlike Sumner et al. [30], we learn the shape space from a *single* model by exploiting part-level symmetries using inter- and intra-part correlations.

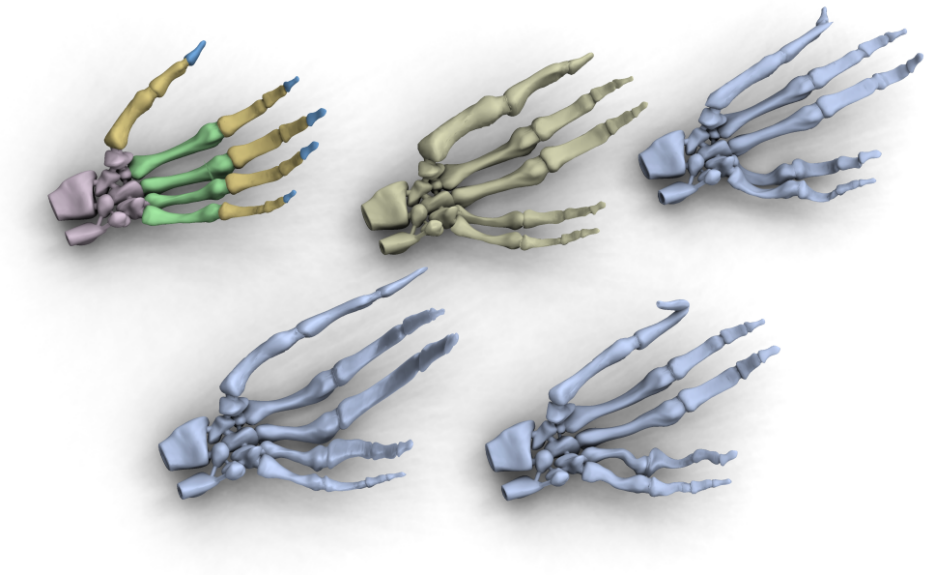


Figure 6.1: *Comparison to standard deformation techniques - clockwise from upper left: original input, our result, Laplacian surface editing, finite-element based elasticity, thin-plate-splines.*

We also compared MPM to a larger set of previously proposed deformation techniques (see Figure 6.1): We tested a finite-element-based elasticity model [1], which behaves similar to the Laplacian surface editing, including all the artifacts. A thin-plate spline deformation model [12] that fits a smooth deformation field rather than minimizing stretch avoids the stretch and wrinkling artifacts, but the

results still look unrealistic. In particular, thin-plate-splines permit arbitrary affine mappings at no cost which leads to distorted results (anisotropic squeezing and expansion).

We also tested discrete modifications to the example models, setting a custom part graph consistent with the established shape grammar (see Figure 4.1). Our solver assembles the pieces together in plausible configurations as if the shapes had been designed like that, even in presence of cyclic dependencies.

Figure 5.2 shows the effect of directly prescribing subspace coordinates for the parts: In Figure 5.2a, subspace coordinates are diffused to their neighbors during editing to encourage a locally symmetric behavior, while Figures 5.2b and 3.3 show global symmetrization. In Figures 5.2b, symmetrization is used interactively, with the mean derived from user constraints. Caricature, the opposite effect is obtained when we increase the distance of the shape parameters from the learned mean (see Figure 3.3e).

Limitations: In our framework, we require manually specified sparse correspondences. While automating this pre-processing stage is desirable, the problem is challenging and an active topic of research [21, 6]. Further, we plan to investigate automatic spawning of new part graphs, which is currently manually specified. Our framework only applies to objects with a clear part structure with one-to-one correspondence: we cannot capture certain types of redundancy such as fractal terrains or irregular textures such as bark of a tree. Finally, in highly complex models, say a detailed car model consists of many parts, a hierarchical decomposition is desirable. Potentially a coarse structure (or cage) can be used in this regard.

7 Conclusions and Future Work

We presented a novel technique for shape deformations based on morphable parts. Instead of using predefined differential shape priors such as elastic (as-rigid-as-possible) or continuous (thin-plate splines) behavior, we learn the variations from correspondences across symmetric parts from a *single* input example. Our model represents shape variations of parts, and pairwise relations of connected parts. With only a sparse set of user constraints, we enable plausible deformations. In contrast to previous techniques based on morphable models, our approach utilizes partial symmetry to extract more information from the input data and learns rich spaces of shape variations from small amounts of training data. We explored applications such as partial symmetrization, caricatures, and presented first steps towards generalized inverse procedural modeling by learning a deformable shape grammar and reassembling parts accordingly.

There are a number of avenues for future work: We belief that by using recent work on learned shape correspondences [21, 6] it should be possible to handle large data bases with minimal user intervention. An interesting future direction is to build and evaluate discrete graphs to perform fully automated data-driven modeling.

Bibliography

[1] B. Adams, M. Ovsjanikov, M. Wand, H.-P. Seidel, and L. J. Guibas. Meshless modeling of deformable shapes and their motion. In *Symposium on Computer Animation*, 2008.

[2] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: reconstruction and parameterization from range scans. In *Proc. of ACM SIGGRAPH*, pages 587–594, 2003.

[3] I. Baran, D. Vlasic, E. Grinspun, and J. Popović. Semantic deformation transfer. *ACM Trans. Graph.*, 28, 2009.

[4] J. Barbic, M. da Silva, and J. Popovic. Deformable object animation using reduced optimal control. *ACM Trans. Graph.*, 28, 2009.

[5] M. Ben-Chen, O. Weber, and C. Gotsman. Variational harmonic maps for space deformation. *ACM Transactions on Graphics*, 28(3), 2009.

[6] A. Berner, M. Wand, N. Mitra, D. Mewes, and H.-P. Seidel. Subspace symmetries. In *Proc. Eurographics*, 2011. to appear.

[7] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proc. SIGGRAPH*, pages 187–194, 1999.

[8] M. Bokeloh, M. Wand, and H.-P. Seidel. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph.*, 29:104:1–104:10, July 2010.

[9] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. Primo: coupled prisms for intuitive surface modeling. In *Symposium on Geometry Processing*, pages 11–20, 2006.

[10] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, 2008.

[11] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[12] B. Brown and S. Rusinkiewicz. Global non-rigid alignment of 3-d scans. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3), 2007.

[13] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[14] R. Davies, C. Twining, T. Cootes, J. Waterton, and C. Taylor. A minimum description length approach to statistical shape modeling. *IEEE Transactions on Medical Imaging*, 21(5):525–537, May 2002.

[15] W.-W. Feng, B. Kim, and Y. Yu. Real-time data-driven deformation using kernel canonical correlation analysis. *ACM Transactions on Graphics*, 27(3), 2008.

[16] R. Gal, O. Sorkine, N. Mitra, and D. Cohen-Or. iwires: An analyze-and-edit approach to shape manipulation. *ACM Trans. Graph.*, 28(3), 2009.

[17] N. Gelfand and L. J. Guibas. Shape segmentation using local slippage analysis. In *Proc. Symp. Geometry processing*, pages 214–223, 2004.

[18] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel. A statistical model of human pose and body shape. *CGF (EUROGRAPHICS)*, 28(2), 2009.

[19] A. Hill and C. J. Taylor. Automatic landmark generation for point distribution models. In *BMV*, pages 429–438, 1994.

[20] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Computer Graphics*, 24(3):1134–1141, 2005.

[21] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3d mesh segmentation and labeling. *ACM Trans. Graph.*, 29(3), 2010.

[22] M. Kilian, N. J. Mitra, and H. Pottmann. Geometric modeling in shape space. *Proc. of ACM SIGGRAPH*, 26(3):#64, 1–8, 2007.

[23] A. C. Kotcheff and C. J. Taylor. Automatic construction of eigenshape models by direct optimization. *Medical Image Analysis*, 2(4):303–314, 1998.

[24] V. Kraevoy, A. Sheffer, A. Shamir, and D. Cohen-Or. Non-homogeneous resizing of complex models. *ACM Trans. Graph.*, 27(5):1–9, 2008.

[25] Y. Lipman, D. Levin, and D. Cohen-Or. Green coordinates. *ACM Trans. Graph.*, 27, August 2008.

[26] H. Pottmann, T. Steiner, M. Hofer, C. Haider, and A. Hanbury. The isophotic metric and its application to feature sensitive morphology on surfaces. In *Proc. Euro. Conf. on Comp. Vis.*, 2004.

[27] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proc. Siggraph*, pages 151–160, 1986.

[28] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 109–116, 2007.

[29] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Symposium on Geometry processing*, pages 175–184, 2004.

[30] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović. Mesh-based inverse kinematics. *Proc. of ACM SIGGRAPH*, 24(3):488–495, 2005.

[31] J. R. Tena, F. De la Torre, and I. Matthews. Interactive region-based linear 3d face models. In *Proc. of ACM SIGGRAPH*, 2011.

[32] M. Wand, P. Jenke, Q.-X. Huang, M. Bokeloh, L. Guibas, and A. Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *Proc. Symp. Geometry Processing*, 2007.

[33] Y. Wang, K. Xu, J. Li, H. Zhang, A. Shamir, L. Liu, Z. Cheng, and Y. Xiong. Symmetry hierarchy of man-made objects. In *Eurographics 2011*, 2011. to appear.

[34] T. Winkler, J. Drieseberg, M. Alexa, and K. Hormann. Multi-scale geometry interpolation. *CGF (EUROGRAPHICS)*, 29(2):309–318, May 2010.

[35] W. Xu, J. Wang, K. Yin, K. Zhou, M. van de Panne, F. Chen, and B. Guo. Joint-aware manipulation of deformable models. *ACM Trans. Graph.*, 28(3):1–9, 2009.

[36] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz. Spacetime faces: High-resolution capture for modeling and animation. In *Proc. of ACM SIGGRAPH*, pages 548–558, 2004.

[37] Y. Zheng, H. Fu, D. Cohen-Or, O. K.-C. Au, and C.-L. Tai. Component-wise controllers for structure-preserving shape manipulation. In *Eurographics 2011*, page to appear, 2011.

Below you find a list of the most recent research reports of the Max-Planck-Institut für Informatik. Most of them are accessible via WWW using the URL `http://www.mpi-inf.mpg.de/reports`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
– Library and Publications –
Campus E 1 4

D-66123 Saarbrücken

E-mail: `library@mpi-inf.mpg.de`

MPI-I-2011-5-002	B. Taneva, M. Kacimi, G. Weikum	Finding images of rare and ambiguous entities
MPI-I-2010-RG1-001	M. Suda, C. Weidenbach, P. Wischnewski	On the saturation of YAGO
MPI-I-2010-5-008	S. Elbassuoni, M. Ramanath, G. Weikum	Query relaxation for entity-relationship search
MPI-I-2010-5-007	J. Hoffart, F.M. Suchanek, K. Berberich, G. Weikum	YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia
MPI-I-2010-5-006	A. Broschart, R. Schenkel	Real-time text queries with tunable term pair indexes
MPI-I-2010-5-005	S. Seufert, S. Bedathur, J. Mestre, G. Weikum	Bonsai: Growing Interesting Small Trees
MPI-I-2010-5-003	A. Anand, S. Bedathur, K. Berberich, R. Schenkel	Efficient temporal keyword queries over versioned text
MPI-I-2010-5-002	M. Theobald, M. Sozio, F. Suchanek, N. Nakashole	URDF: Efficient Reasoning in Uncertain RDF Knowledge Bases with Soft and Hard Rules
MPI-I-2010-5-001	K. Berberich, S. Bedathur, O. Alonso, G. Weikum	A language modeling approach for temporal information needs
MPI-I-2010-1-001	C. Huang, T. Kavitha	Maximum cfardinality popular matchings in strict two-sided preference lists
MPI-I-2009-RG1-005	M. Horbach, C. Weidenbach	Superposition for fixed domains
MPI-I-2009-RG1-004	M. Horbach, C. Weidenbach	Decidability results for saturation-based model building
MPI-I-2009-RG1-002	P. Wischnewski, C. Weidenbach	Contextual rewriting
MPI-I-2009-RG1-001	M. Horbach, C. Weidenbach	Deciding the inductive validity of $\forall\exists^*$ queries
MPI-I-2009-5-007	G. Kasneci, G. Weikum, S. Elbassuoni	MING: Mining Informative Entity-Relationship Subgraphs
MPI-I-2009-5-006	S. Bedathur, K. Berberich, J. Dittrich, N. Mamoulis, G. Weikum	Scalable phrase mining for ad-hoc text analytics
MPI-I-2009-5-005	G. de Melo, G. Weikum	Towards a Universal Wordnet by learning from combined evidenc
MPI-I-2009-5-004	N. Preda, F.M. Suchanek, G. Kasneci, T. Neumann, G. Weikum	Coupling knowledge bases and web services for active knowledge
MPI-I-2009-5-003	T. Neumann, G. Weikum	The RDF-3X engine for scalable management of RDF data
MPI-I-2009-5-002	M. Ramanath, K.S. Kumar, G. Ifrim	Generating concise and readable summaries of XML documents

MPI-I-2009-4-006	C. Stoll	Optical reconstruction of detailed animatable human body models
MPI-I-2009-4-005	A. Berner, M. Bokeloh, M. Wand, A. Schilling, H. Seidel	Generalized intrinsic symmetry detection
MPI-I-2009-4-004	V. Havran, J. Zajac, J. Drahokoupil, H. Seidel	MPI Informatics building model as data for your research
MPI-I-2009-4-003	M. Fuchs, T. Chen, O. Wang, R. Raskar, H.P.A. Lensch, H. Seidel	A shaped temporal filter camera
MPI-I-2009-4-002	A. Tevs, M. Wand, I. Ihrke, H. Seidel	A Bayesian approach to manifold topology reconstruction
MPI-I-2009-4-001	M.B. Hullin, B. Ajdin, J. Hanika, H. Seidel, J. Kautz, H.P.A. Lensch	Acquisition and analysis of bispectral bidirectional reflectance distribution functions
MPI-I-2008-RG1-001	A. Fietzke, C. Weidenbach	Labelled splitting
MPI-I-2008-5-004	F. Suchanek, M. Sozio, G. Weikum	SOFI: a self-organizing framework for information extraction
MPI-I-2008-5-003	G. de Melo, F.M. Suchanek, A. Pease	Integrating Yago into the suggested upper merged ontology
MPI-I-2008-5-002	T. Neumann, G. Moerkotte	Single phase construction of optimal DAG-structured QEPs
MPI-I-2008-5-001	G. Kasneci, M. Ramanath, M. Sozio, F.M. Suchanek, G. Weikum	STAR: Steiner tree approximation in relationship-graphs
MPI-I-2008-4-003	T. Schultz, H. Theisel, H. Seidel	Crease surfaces: from theory to extraction and application to diffusion tensor MRI
MPI-I-2008-4-002	D. Wang, A. Belyaev, W. Saleem, H. Seidel	Estimating complexity of 3D shapes using view similarity
MPI-I-2008-1-001	D. Ajwani, I. Malinger, U. Meyer, S. Toledo	Characterizing the performance of Flash memory storage devices and its impact on algorithm design
MPI-I-2007-RG1-002	T. Hillenbrand, C. Weidenbach	Superposition for finite domains
MPI-I-2007-5-003	F.M. Suchanek, G. Kasneci, G. Weikum	Yago : a large ontology from Wikipedia and WordNet
MPI-I-2007-5-002	K. Berberich, S. Bedathur, T. Neumann, G. Weikum	A time machine for text search
MPI-I-2007-5-001	G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum	NAGA: searching and ranking knowledge
MPI-I-2007-4-008	J. Gall, T. Brox, B. Rosenhahn, H. Seidel	Global stochastic optimization for robust and accurate human motion capture
MPI-I-2007-4-007	R. Herzog, V. Havran, K. Myszkowski, H. Seidel	Global illumination using photon ray splatting
MPI-I-2007-4-006	C. Dyken, G. Ziegler, C. Theobalt, H. Seidel	GPU marching cubes on shader model 3.0 and 4.0
MPI-I-2007-4-005	T. Schultz, J. Weickert, H. Seidel	A higher-order structure tensor
MPI-I-2007-4-004	C. Stoll, E. de Aguiar, C. Theobalt, H. Seidel	A volumetric approach to interactive shape editing
MPI-I-2007-4-003	R. Bargmann, V. Blanz, H. Seidel	A nonlinear viseme model for triphone-based speech synthesis
MPI-I-2007-4-002	T. Langer, H. Seidel	Construction of smooth maps with mean value coordinates
MPI-I-2007-4-001	J. Gall, B. Rosenhahn, H. Seidel	Clustered stochastic optimization for object recognition and pose estimation

MPI-I-2007-2-001	A. Podelski, S. Wagner	A method and a tool for automatic veriication of region stability for hybrid systems
MPI-I-2007-1-003	A. Gidenstam, M. Papatriantafilou	LFthreads: a lock-free thread library
MPI-I-2007-1-002	E. Althaus, S. Canzar	A Lagrangian relaxation approach for the multiple sequence alignment problem
MPI-I-2007-1-001	E. Berberich, L. Kettner	Linear-time reordering in a sweep-line algorithm for algebraic curves intersecting in a common point
MPI-I-2006-5-006	G. Kasnec, F.M. Suchanek, G. Weikum	Yago - a core of semantic knowledge
MPI-I-2006-5-005	R. Angelova, S. Siersdorfer	A neighborhood-based approach for clustering of linked document collections
MPI-I-2006-5-004	F. Suchanek, G. Ifrim, G. Weikum	Combining linguistic and statistical analysis to extract relations from web documents
MPI-I-2006-5-003	V. Scholz, M. Magnor	Garment texture editing in monocular video sequences based on color-coded printing patterns
MPI-I-2006-5-002	H. Bast, D. Majumdar, R. Schenkel, M. Theobald, G. Weikum	IO-Top-k: index-access optimized top-k query processing