EUROGRAPHICS 2011 / M. Chen and O. Deussen
(Guest Editors)

Volume 30 (2011), Number 2

Learning Line Features in 3D Geometry

M. Sunkel[1] S. Jansen[1] M.Wand[1,2] E.Eisemann[3] H.-P. Seidel[1]

[1]Max-Planck Institute Informatik [2]Saarland University [3]Telecom ParisTech/CNRS-LTCI

Abstract

Feature detection in geometric datasets is a fundamental tool for solving shape matching problems such as partial symmetry detection. Traditional techniques usually employ a priori models such as crease lines that are unspecific to the actual application. Our paper examines the idea of learning geometric features. We introduce a formal model for a class of linear feature constellations based on a Markov chain model and propose a novel, efficient algorithm for detecting a large number of features simultaneously. After a short user-guided training stage, in which one or a few example lines are sketched directly onto the input data, our algorithm automatically finds all pieces of geometry similar to the marked areas. In particular, the algorithm is able recognize larger classes of semantically similar but geometrically varying features, which is very difficult using unsupervised techniques. In a number of experiments, we apply our technique to point cloud data from 3D scanners. The algorithm is able to detect features with very low rates of false positives and negatives and to recognize broader classes of similar geometry (such as "windows" in a building scan) even from few training examples, thereby significantly improving over previous unsupervised techniques.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Artificial Intelligence [I.2.10]: Vision and Scene Understanding—

1. Introduction

Feature matching is an important tool in geometry processing to address correspondence problems such as rigid and deformable shape matching [ASP*04, GMGP05, HFG*06, BBK06, LG05], or symmetry detection [MGP06, PMW*08, BBW*09b]. Its principle is to find discrete objects that describe characteristic pieces of the input which is useful in many contexts: cleanup and precision improvements of scanned data, auto completion and hole filling, meshing, instance highlighting and many more. With the recent and growing wealth of available scanned and modeled geometry, such tasks become increasingly important.

Almost all algorithms for global correspondences (without known initialization) start by detecting geometric features that are invariant under the considered correspondence transformation. Then, consistent correspondences are found via global optimization techniques such as branch and bound, loopy belief propagation, or Hough-transform-based voting. Afterwards, dense correspondences can be estimated using local optimization techniques such as gradient descent (to refine transformation parameters) or region growing (to refine the corresponding area in partial matching scenarios). The first step, to detect invariant features, is crucial in such a pipeline, as it determines the type of correspondences to look for. If no features are detected for a part of the geometry, it cannot be recognized. Similarly, having too many fea-

Figure 1: *We propose an interactive learning algorithm for finding features that are similar to user strokes. In this example, 2-3 examples curves have been sketched on the model for each class (indicated by the color), further instances are found automatically.*

tures leads to clutter that might lead to failures too. From this point of view, traditional feature detection algorithms (based on a fixed model) have some shortcomings:

- **Application specific:** A fixed model determines a priori what constitutes an interesting feature. Different data characteristics may require designing a new detector.
- **Unspecific response:** The detector might output a large number of elements, while only a fraction corresponds to structures we aimed for, putting an additional burden on later matching stages.
- **Lack of generalization:** Traditional feature detectors and descriptors cannot express classes of semantically similar but geometrically different features by the same model.

For some application domains, such as rigid alignment of 3D objects or fundamental matrix estimation (structure from motion) in images, these shortcomings are acceptable because these problems are well-constrained by a small number of correct matches. Here, state-of-the-art techniques such as scale-space blob detection via SIFT-like algorithms have become standard "out-of-the-box" tools [Low03, LG05]. However, there are many upcoming shape-matching problems that aim at a deeper "shape understanding" and make a large number of feature matches necessary. A typical example is partial symmetry detection where objects are decomposed into building blocks [MGP06,PSG*06,PMW*08] and potentially reassembled to form new models [BWS10]. Here, the design of the feature detector is often the limiting factor in the recognition process [BBW*08, BBW*09b]. In extension to this, there is the much harder problem of recognizing semantically similar but geometrically different parts which is a mostly unsolved problem [ZSCO*08, BBW*09a, ATCO*10] and seems to push fixed-priori-model approaches to their limits. In this paper, we propose to get the user into the loop to better address these issues by learning a matching model. The user can specify what to look for using strokes (e.g., windows or doors in a 3D scan).

Our choice to use strokes follows previous findings [OBS04, HPW05, BBW*09b] for unsupervised settings which indicate that crease lines have advantages in characterizing salient features in 3D geometry. Line features are numerous, but more discriminative than just keypoints and their locally adjacent regions. We develop a formal framework to describe such features based on Markov chains. It combines two desirable properties: It is expressive enough to learn complex feature lines and their variations. Furthermore, the solution can be computed efficiently: We present a novel algorithm to solve the interference problem within seconds by computing all significant local optima simultaneously, dramatically improving over iterated naïve search.

Our model is particularly designed to run with minimal and simple user input, which is crucial for making it a useful interactive tool. Typically, a single stroke is enough to train simple detectors, while a few can span more complex classes with significant geometric variation. Our weakly supervised approach consistently outperforms complex unsupervised algorithms. In particular, we can detect semantically similar parts which permits applications currently mostly inaccessible to unsupervised global correspondence algorithms.

2. Related Work

The goal of our technique is to establish correspondences among semantically similar geometric features. There is a large body of work on correspondence estimation for geometric data sets. Early techniques such as the classic iterated closest point (ICP) algorithm for rigid matching [BM92, CM92] as well as the later deformable ICP techniques [ACP03, HTB03, BBK06, BR07] are based on local optimization, thus requiring a user-guided initialization.

More recently, several global optimization techniques have been proposed that solve the problem without prealignment [ASP*04, GMGP05, LG05, HFG*06, CZ08, HAWG08, ZSCO*08, TBW*09]. An interesting variant is partial symmetry detection, where a single shape is decomposed into building blocks. A common approach is transformations voting, which detects symmetries under a fixed group of transformations such as similarity transformations [MGP06, PSG*06, PMW*08]. The use of a location independent voting space can, however, lead to problems if many symmetric parts are present simultaneously. Matching graphs of surface feature [BBW*08] has been proposed to avoid this problem. In particular, matching configurations of crease-line features [BBW*09b] have recently demonstrated good performance in many practical applications and this is one of the main motivations for our work. Feature-based matching, however, comes at a cost, which is the reliance on feature detection. Obviously, a fixed detection scheme is not suitable for all kinds of data characteristics. With our user-guided model, it is possible to interactively train detectors for different application scenarios without creating a new matching model from scratch. The limitations of a fixed-model become also apparent when trying to match dissimilar but semantically related geometry [BBW*09a, ZSCO*08, ATCO*10]. This problem is notoriously ill-posed. A small amount of user-trained knowledge can help significantly in making this problem conceptually as well as computationally feasible.

Learning of feature matching is rare in geometry processing. Schoelkopf et al. [SSB05] apply regression techniques from machine learning to estimate correspondences but do not learn from user input. In follow-up work, Steinke et al. [SSB06] identify invariant feature regions in a morphable face model [BV99] by an oriented principal component analysis (PCA). Their method requires some amount of training data and outputs a weighting function that describes the invariance of local descriptors. In contrast, we are aiming at finding sparse and discrete feature lines from minimal user input. A recent paper of Kalogerakis et al. [KHS10] explored learning user segmentation by example. It uses, somewhat

similar to our approach, a (loopy, in their case) Markov random field (MRF) with learned local descriptors at nodes and learned edge compatibility costs. Then, MAP estimation is performed using iterated graph cuts. Unlike our approach, no correspondences are computed within the segmented regions, and only one global solution is computed.

User-labeled images have been used extensively in computer vision for object recognition, automatic segmentation, and image understanding [CET01,VJ01,LLS04,FH05]. An important ingredient are classifiers with strong generalization capabilities, such as "face" recognition in images [VJ01]. In our setting, we are able to use simpler local classification techniques within our model because geometry is far less ambiguous than images that suffer from lighting, shading, projection, and occlusion differences. As global models, subspace (PCA) techniques have been used [CET01, BV99] which are very expressive but difficult to globally optimize, as well as generalized voting techniques (e.g. [LLS04]). The technique most similar to our approach is Felzenszwalb et al.'s "Pictorial Structures" that use a tree structure MRF of local appearances to describe objects by decomposition into parts and spatial relations. While conceptually related, our setting differs in two important aspects: First, we are dealing with general 2-manifolds in an irregularly sampled representation (point clouds) as inference domain which provides more degrees of freedom and is more difficult to handle than regular pixel grids. Secondly, unlike in image understanding applications, we are aiming at finding many instances simultaneously rather than only the best explanation for an image.

3. Matching Model

The goal of our technique is to recognize curves on surfaces that coincide with geometric features. Our current implementation works on surfaces approximated by finite point clouds, therefore being applicable to raw 3D scanner data. An implementation for other representations (e.g., triangle meshes) would require only minimal modifications. Curves are represented as polygonal lines. Training data is expected to be given in the same format, with nodes being placed at corresponding locations. Our goal is to learn the geometric characteristics of these curves and subsequently find all occurrences of similar geometry in an input data set.

3.1. Feature Lines

Formally, assume that we are given an input surface $\mathcal{S} \subset \mathbb{R}^3$ that is a 2-manifold of arbitrary topology. From the perspective of our algorithm, it will be given a sampled representation $\tilde{\mathcal{S}} = \{\mathbf{s}_1, ..., \mathbf{s}_n\}, \mathbf{s}_i \in \mathbb{R}^3$. We encode feature lines on $\mathcal{S}$ via piecewise linear curves (polylines) $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_k\}$ where each $\mathbf{x}_i \in \tilde{\mathcal{S}}$ is one corner point along the curve.

One specific class of feature lines $\mathcal{F}$ is characterized by statistics on both the shape of the curve $\mathbf{X}$ as well as statistics

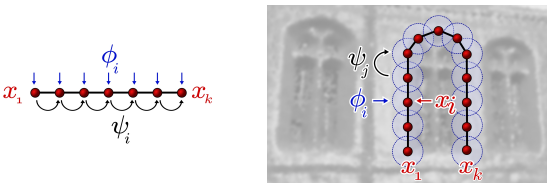

Figure 2: *Feature lines are modeled as Markov chains: Each node is controlled by a potential function ϕ that matches local geometry. In addition, pairwise potentials ψ encode the shape variation of the curve.*

on the underlying geometry. A class of feature lines is characterized by a probability density function $p : \tilde{\mathcal{S}}^k \to \mathbb{R}^{\geq 0}$. We describe the feature line by a Markov chain model: The probability density for a polyline is described by singleton potential functions ϕ_i that prescribe local geometry and by pairwise potential functions ψ_i that encode the shape of the curve itself ($1/Z$ is the normalization constant):

$$p(\mathbf{X} \in \mathcal{F}) = \frac{1}{Z} \prod_{i=1}^{k} \phi_i(\mathbf{x}_i) \prod_{i=1}^{k-1} \psi_i(\mathbf{x}_i, \mathbf{x}_{i+1}) \qquad (1)$$

This means that a class of feature lines is completely characterized by specifying a set of potential functions $\{(\phi_i)_i, (\psi_i)_i\}$. In the following section, we will look at how to define and learn these functions from user input.

4. Learning

For representing both the $\{(\phi_i)_i$ and the $(\psi_i)_i\}$, we use simple Gaussian models that encode low-dimensional subspaces in the shape space. Obviously, there are much more sophisticated learning techniques such as support vector machines or boosted classifiers that are frequently applied at this point in computer vision applications [VJ01, PJC*10]. However, we are aiming at learning from a minimal amount of training data (often only one or two examples). Complex non-linear methods are of little use here as the training data itself does not support many degrees of freedom. Thus, a simple linear learning technique is an adequate choice in our scenario.

To simplify, we assume that all models have a designated upward direction. For models such as architectural scenes, statues, as well as many other man-made and natural artifacts, this is the case and can be easily specified interactively. We will denote this direction as **up** in the following. In conjunction with the surface normal $\mathbf{n}(\mathbf{x}), \mathbf{x} \in \tilde{\mathcal{S}}$, this gives us a unique coordinate frame for each surface point, removing a rotational degree of freedom from the matching. In the following, we use $(\mathbf{u}(\mathbf{x}), \mathbf{v}(\mathbf{x}), \mathbf{n}(\mathbf{x}))$, or in short $(\mathbf{u}, \mathbf{v}, \mathbf{n})$ when clear from the context, to denote an orthogonal, right-handed coordinate frame where $\mathbf{n}$ is the surface normal and $\mathbf{u}$ is a vector orthogonal to $\mathbf{n}$ that is closest to **up**.

User Interaction: For the learning phase, we ask the user to sketch one or more polylines on the surface. We have imple-

mented a tool that combines a 3D point-cloud viewer with a surface curve editor to specify such curves interactively. Each click leaves a control point on the surface, which makes the curve initialization very simple. If multiple curves are specified for a model, our algorithm expects the user to put nodes of the curve in corresponding positions. This is not a big problem in practice as users usually tend to chose salient features as node points that are rather easy to locate.

Learning Local Evidence: We define the density $\phi_i(\mathbf{x})$ as a Gaussian model in the shape space of local pieces of geometry in the vicinity of each data point: We cutout consistent regions around each point of the model and form local depth images along the normal axis. We will refer to these as *descriptors*. The descriptor space is high dimensional: every depth image pixel represents one degree of freedom. We learn a low dimensional subspace as Gaussian distribution by a PCA of corresponding training points. For a single training example, this automatically degrades to learning a constant mean depth image.

Technically, we parameterize the neighborhood of each point $\mathbf{x} \in \tilde{\mathcal{S}}$ as local heightfields [PG01]: We fix a radius parameter r_ϕ, specified by the user. It specifies the "thickness" of the feature line, i.e., how far the algorithm should reach out around each node to compare local geometry. We then cut out all points within a radius of r_ϕ from $\mathbf{x}$ and project them in normal direction onto the tangent plane in $\mathbf{x}$ spanned by $(\mathbf{u},\mathbf{v})$. Let (u_j, v_j, n_j) denote the respective coordinates of the points in the local tangent frame. We collect all points that fall within a rectangular region $u_j, v_j \in [-r_\phi, ..., r_\phi]$ and store their normal direction n_j. We finally splat these values into a $d \times d$ pixel grid (we use a fixed $d = 16$) and use a push-pull algorithm to fill potential holes with a smooth interpolation. We precompute these descriptors for all points $\mathbf{x} \in \tilde{\mathcal{S}}$ in the input and compress them to more compact d_c-dimensional vectors using PCA (in our experiments, we keep the $d_c = 16$ dominant of 256 dimensions of the PCA). This reduces computational costs but it will not significantly reduce the discriminability of the descriptors because the projection will well preserve distances in the descriptor space [DG03]. We denote these values $\mathbf{d_x}$ (for all $\mathbf{x} \in \tilde{\mathcal{S}}$).

Given this database of local geometry, learning a Gaussian model of $\phi_i(\mathbf{x})$ is straightforward. Assume that the user has specified m input curves, each consisting of k nodes $\mathbf{x}_i^{(j)}, i = 1..k, j = 1..m$. We compute a d_c-dimensional mean $\mu_i^{(\phi)}$ and a $d_c \times d_c$ covariance matrix $\Sigma_i^{(\phi)}$ of the sets $\{\mathbf{d}_{\mathbf{x}_i^{(j)}}\}_j$ for each node $i = 1..k$ across the curves using PCA. The so-defined subspaces of significant variance will typically be much lower dimensional than the original descriptor space because they were computed from only very few and presumably highly correlated examples. We store these statistics to characterize the training set. We then evaluate ϕ_i as multi-variate normal distribution:

$$\phi_i(\mathbf{x}) \sim \exp(-\frac{1}{2}(\mathbf{d_x} - \mu_i^{(\phi)})^T \overline{\Sigma^{(\phi)}}_i^{-1} (\mathbf{d_x} - \mu_i^{(\phi)})) \quad (2)$$

where

$$\overline{\Sigma^{(\phi)}}_i := \Sigma_i^{(\phi)} + \lambda \mathbf{I}. \quad (3)$$

Here, we do not use the original covariance matrices but add an additional uniform noise term, specified by the user parameter λ. This term models uniform Gaussian noise in the data. Typically, it should be set according to the measurement accuracy. Often, this is not known exactly and has to be estimated roughly. Increasing this parameter also increases the tolerance towards shape variations that are not captured in the training set. In the user interface, the parameter λ can be specified as fraction of the overall variance (maximum eigenvalue of the covariance matrix of the complete descriptor space) of the input patches. This choice facilitates the definition of λ as it becomes independent of scaling and to some extent invariant to different input data sets. Please note that λ is not learned automatically from the user-specified examples. Using only few examples might not be sufficient to estimate full rank covariance matrices $\Sigma_i^{(\phi)}$, and a statistically reliable estimation of the full matrix would require prohibitively large amounts of training data. Our simple mixture model avoids this problem by separating base the noise floor from one or two main directions of variation in the descriptor space.

Learning the Curve Shape: Learning the curve parameters proceeds analogously: For each user specified curve, we measure the length and the *twist* of each curve segment $\{\mathbf{x}_i, \mathbf{x}_{i+1}\}$. The length is just the Euclidean distance of the endpoints. The twist is a vector of two angles that describes how to rotate around the normal $\mathbf{n}(\mathbf{x})$ in order to align the new with the previous curve segment, thereby encoding the orientation. For each triple of length and or orientation parameters, we again learn means μ_i^Ψ and 3×3 covariance matrices Σ_i^Ψ via a PCA analysis of corresponding curve segments in the user-defined training set. We scale the length and the two angle parameters with three constant factors to obtain a unit variance for each curve segment over all curves. This whitening transform is necessary because angles and length are measured in different units so that the main axis of the PCA might be affected by scaling rather than data characteristics. In order to form the final probability densities $\psi_i(\mathbf{x},\mathbf{y})$, we again add isotropic Gaussian noise to account for general inaccuracies. In the following, let $\mathbf{c_{x,y}}$ denote the scaled length and orientation parameters. Using this notation, we obtain:

$$\psi_i(\mathbf{x},\mathbf{y}) \sim \exp(-\frac{1}{2}(\mathbf{c_{x,y}} - \mu_i^\Psi)^T \overline{\Sigma^\Psi}_i^{-1} (\mathbf{c_{x,y}} - \mu_i^\Psi)) \quad (4)$$

with

$$\overline{\Sigma^\Psi}_i := \Sigma_i^\Psi + \begin{pmatrix} \lambda_{length}^2 & 0 & 0 \\ 0 & \lambda_{angle}^2 & 0 \\ 0 & 0 & \lambda_{angle}^2 \end{pmatrix}. \quad (5)$$

The parameters λ_{length} and λ_{angle} describe the anticipated variation of edge length and orientation. In the user interface, length variations are specified relative to the average segment length (in percent) and angular variation by the expected standard deviation in degrees.

Furthermore, we add an option to scale all covariance matrices that are learned from data (for both local evidence and curve shape) by a factor > 1 so that more variation in the learned direction is allowed. This is useful as small sample sets might not capture the full variance.

5. Inference

Our task is now to find all solutions that match our probabilistic model (Eq. 1). The technical challenge is that Equation 1 defines a probability density on a high-dimensional space: For k curve points, we have to consider $|\tilde{\mathcal{S}}|^k$ different assignments. A brute-force evaluation is clearly infeasible. Therefore, we first reduce the distribution to a polynomially sized representation and then extract the solutions from that representation, similar to [LTSW09].

As a reduced representation, we use the max-marginals ν_i of p with respect to fixed node positions $\mathbf{x}_i = \mathbf{y}$:

$$\nu_i(\mathbf{y}) = \max_{\substack{\mathbf{x}\in\tilde{\mathcal{S}}^k \\ \text{with } \mathbf{x}_i=\mathbf{y}}} p(\mathbf{x}_1,..,\mathbf{x}_i,..,\mathbf{x}_k) \tag{6}$$

This means, all curves that go through $\mathbf{y}$ at node i are considered and the maximum probability value is kept.

Explicitly computing the maximum in Eq. 6 by enumeration is exponential as well. However, it is well known that for tree structured MRFs (and thus for chains in particular), the computation can be performed in polynomial time using dynamic programming. The max-product belief propagation algorithm compute exact marginals in $\mathcal{O}(kn^2)$ time for k nodes (curve nodes in our case) and n states (discrete points in $\tilde{\mathcal{S}}$ in our case) [YFW01]. We will now first briefly recap how belief propagation is traditionally used to compute a single globally optimal solution (Subsec. 5.1). Afterwards, we describe our modified algorithm that finds several locally optimal solutions simultaneously (Subsec. 5.2). Afterwards, we describe how we augment the algorithm to avoid the $\mathcal{O}(n^2)$ complexity (Subsec. 5.3).

5.1. Belief Propagation

Belief propagation on chains works in two passes [YFW01]: First, information is propagated from the start node through all nodes of the chain until the end node. The end node then has gathered enough information to compute the correct max-marginals. In a second pass, information is propagated back to the start to obtain correct max-marginals everywhere. The messages combine knowledge from the local evidence term ϕ, the compatibility term ψ and the incoming belief about the state of the node:

$$m^{\text{fwd}}_{i\to i+1}(\mathbf{x}_{i+1}) = \max_{\mathbf{x}_i} \phi_i(\mathbf{x}_i)\psi_i(\mathbf{x}_i,\mathbf{x}_{i+1})m^{\text{fwd}}_{i-1\to i}(\mathbf{x}_i)\,, \tag{7}$$

where $m^{\text{fwd}}_{0\to 1}(\mathbf{x}) = 1$. The backward messages $m^{bwd}_{i\to i-1}$ are constructed analogously, just passing information in the opposite direction. The max-marginals are afterwards given by

$$\nu_i(\mathbf{x}_i) = \phi_i(\mathbf{x}_i)m^{\text{fwd}}_{i-1\to i}(\mathbf{x}_i)m^{\text{bwd}}_{i+1\to i}(\mathbf{x}_i). \tag{8}$$

If we want to compute the globally optimal solution, we have to perform backtracking on the solution, which can be conveniently combined with the backward message passing, where each step yields the needed correct max-marginal at that node: We start backtracking from the k-th node. We output a maximum likelihood curve $\mathbf{Z} = \{\mathbf{z}_1,\ldots,\mathbf{z}_k\}$ by choosing the maximum of the max-marginal of node k and then combining this solution with the best match at $k-1$ and then iterating backwards:

$$\mathbf{z}_k = \arg\max_{\mathbf{x}_k} \nu_k(\mathbf{x}_k); \quad \mathbf{z}_i = \arg\max_{\mathbf{x}_i} \phi_i(\mathbf{x}_i)\psi_i(\mathbf{x}_i,\mathbf{z}_{i+1}) \tag{9}$$

5.2. Computing Several Local Optima Simultaneously

In order to detect a large number of features (which is particularly useful for symmetry detection applications), having only one output curve is not enough. We therefore augment the standard algorithm as follows:

Instead of starting backtracking at a single point $\mathbf{z}_k$ of maximum max-marginal probability, we extract a solution for each local maxima of the distribution ν_k. The restriction to local maxima is neccesary since otherwise (when starting backtracking for all positions $\mathbf{x}_k$ with a $\nu_k(\mathbf{x}_k)$ over a threshold) we would obtain very similar curves with just minimally perturbed positions of the last node. Therefore, we only accept maxima that are the largest value within a window of radius $2r_\phi$ around each point (where r_ϕ is again the descriptor radius). We then start backtracking at each of these solutions. This will extract all at most $\mathcal{O}(n/r_\phi^2)$ locally optimal solutions for which the distance at node $\mathbf{x}_k$ is at least $2r_\phi$.

In cases where solutions overlap at the end node, it is possible to miss locally optimal curves that could diverge in two or more different directions (think of detecting the boundary of a single leaf of a four leaf clover; only one curve will be found if the center is the end node). We can fix this problem by starting the local maxima backtracking for each node of the chain (after computing the max-marginals). We obtain an arrangement of overlaid solution curves that can be combined arbitrarily by cutting segments between crossings of the curves and recombining. In practice, however, such situations are rare; we did not observe this ambiguity in our experiments (even if it did happen, as for the leaf example, it could usually be addressed by changing the start point). We therefore restrict our practical implementation to a single backtracking pass.

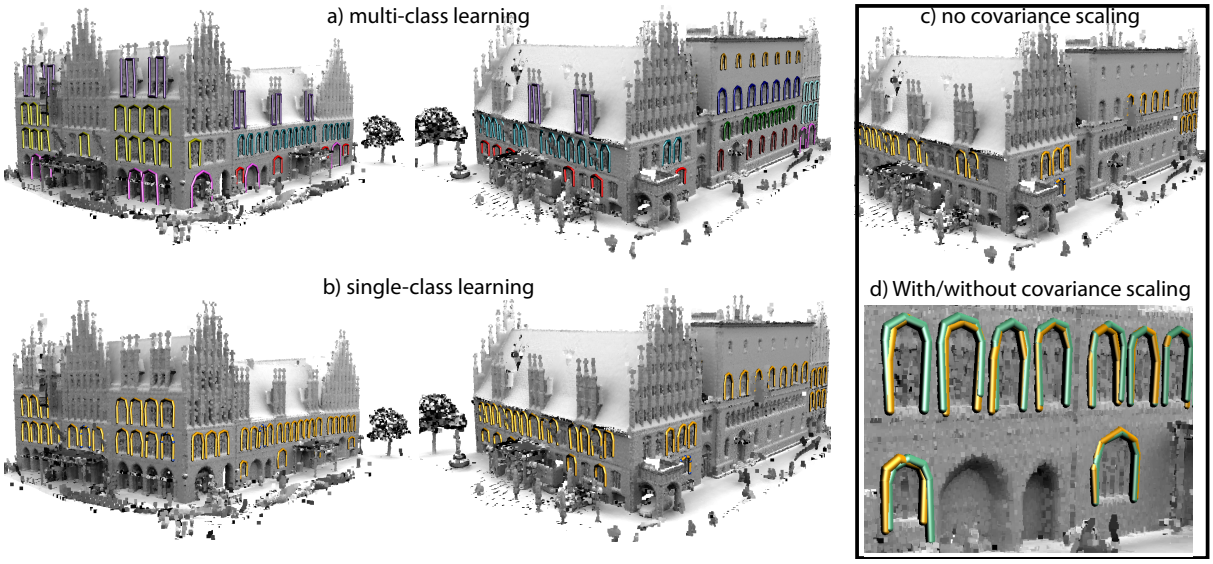

Figure 3: *a) Multi-class learning: Decomposing the old town hall into partially symmetric elements. Up to two examples are used per class. b) Single-class learning: Detecting "all windows". Training data (in blue) consists of only 4 feature lines. c) Without scaling the learned covariance matrices, results deteriorate. (d) Detail of overlay: green (unscaled), yellow (scaled). When using scaled versions of the learned covariance matrices, the extracted curves adopt better to the underlying geometry.*

5.3. Reducing the complexity

So far, our algorithm is too slow in practice: In order to compute a single message in Eq. 7, we have to consider all states of node $\mathbf{x}_{i+1}$ and for each maximize over all possible states $\mathbf{x}_i$, requiring $\mathcal{O}(n^2)$ computations where n is the number of discretization points (typically in the range of millions). This has to be repeated at least $k-1$ times to compute ν_k.

The computational effort can be reduced significantly by taking the properties of the potentials ϕ and ψ into account. We know that ψ_i describes a line segment originating at position $\mathbf{x}_i$ with a length that is constrained by a Gaussian distribution. In addition, the descriptors ϕ are typically non-matching for large portions of the data.

We first replace the Gaussian distributions ϕ_i, ψ_i by truncated versions, were we clamp the value to zero if the density falls below 5% of the maximum. This has the additional advantage of avoiding spurious local extrema of very low density so that we can run the previously described peak extraction algorithm without need for an additional threshold value to filter out promising local extrema.

Given this truncated potentials, we can now restrict the message passing computations: First, in the loop over possible positions $\mathbf{x}_i \in \tilde{\mathcal{S}}$, we only consider points for which the messages computed so far are non-zero. In the first iteration, we consider the evidence term ϕ_1 to make this decision. Secondly, we do not try to combine it with all points $\mathbf{x}_{i+1} \in \tilde{\mathcal{S}}$ to evaluate $\psi(\mathbf{x}_i, \mathbf{x}_{i+1})$ but restrict ourselves to points $\mathbf{x}_{i+1}$ for which $\psi(\mathbf{x}_i, \mathbf{x}_{i+1})$ is potentially non-zero. In order to estimate this, we compute the maximum length l_{max} of the distance $\|\mathbf{x}_i - \mathbf{x}_{i+1}\|$ for which $\psi(\mathbf{x}_i, \mathbf{x}_{i+1})$ is non-zero. This can be achieved by looking at the covariance matrix Σ_i^{ψ} (which correlates angels and distances) and extracting the maximum variance in distance direction. Having l_{max}, we now extract only points $\mathbf{x}_{i+1}$ from $\tilde{\mathcal{S}}$ within that are located within a sphere of radius l_{max} around $\mathbf{x}_i$. We retrieve these points efficiently by using a hierarchical range query. For this, we precompute an octree for the points $\tilde{\mathcal{S}}$. During runtime, we traverse the octree top down, only following boxes that overlap with the spherical range and outputting the points in the leaf nodes that are within the range. In typical applications, the maximum radius l_{max} is very small in comparison to the size of the complete scene so that we get very significant speedups by this optimization.

6. Results and Implementation

We have implemented the described feature detection framework in plain, single threaded C++ and performed experiments on an Intel Core-2 Quad 2.4GHz PC with 8GB main memory. In order to realistically evaluate the performance in practice, we apply our technique to a number of LIDAR rangescans (and two synthetic test scenes, Fig. 10). We use the raw data without any preprocessing other than a simple subsampling for the very dense point clouds. In particular, no smoothing, hole-filling, or outlier detection has been performed. Four of our benchmark data sets are taken from the well-known Hannover city scan collection (available on-

model	#points	#preproc. [s]
old town hall	744,567	516
new town hall	1,271,777	908
museum	2,119,367	3,741
baroque church	2,618,385	2,432
stars	313,236	326
relief	289,509	336
street (training/all)	198,767/1,056,158	436/999

Table 1: *Statistics for our example scenes. Timings are for descriptor radius* 0.01.

line at http://www.ikg.uni-hannover.de). In addition, we use a scan of the "Ludwigskirche", a baroque church from the 18th century, provided by the LKVK Saarland. The LIDAR data suffers from significant noise artifacts. In particular, reflective surfaces such as windows create local clouds of structured, non-Gaussian noise artifacts. The scan quality of the church data set is slightly better, probably due to more modern acquisition equipment.

In the following, we conduct a number of experiments: For each data set, we have annotated structural elements. Afterwards, we use our algorithm to identify the learned classes. We perform different types of experiments: In multiple class learning, our objective is to find elements of different categories. In single class learning, we try to build a single general class for semantically similar objects. In a third experiment type, we perform multiple class learning and restrict ourselves to only a single example per category to study how far we can get with an absolute minimum of user supervision.

Figure 3a shows the result of a **multi-class learning** experiment performed on the Hannover "old town hall" data set. We train separate line feature classes for several types of windows, the small roof towers, and archways. The detected structures include rounded, pointy, and even complex shapes. For each class, we use at most two examples, sketched by the user directly onto the model (as shown in the accompanying video). By setting conservative global variance parameters per class, we can avoid any false positives: No wrong matches are present and members of different classes are not mixed up. At the same time, we obtain a false negative rate of 24% (32 out of 133). Unrecognized elements include several severely distorted pieces such as elements with large scale acquisition holes or strong clutter due to outliers.

We repeat the multi-class experiment on the new town hall data set, obtaining roughly comparable results (see Figure 5). We perform another multi-clss experiment on the museum data set (Figure 6). In this case, we use exactly the same isotropic noise parameters for all classes, nevertheless obtaining good matching results. Figure 7 shows the results for the baroque church. Here, we perform multi-class learning with only a single instance specified per class. This reduces the recognition results a bit, but we nevertheless retrieve more than 70% correctly and do not observe false negatives. Figure 1 shows results of another interactive session where 2 examples per class are used (the round windows marked in green use three examples capturing different types of empty/solid/grid-structured interior).

Next, we perform a **single-class learning experiment**, where a set of windows are comprised in one class. The results are showing in Figure 3b and the variation in the input is illustrated in Figure 4. In this case, we recognize all but 7 out of 84 windows in the model. Out of the 7, three contain large holes such that a recognition is obviously impossible. Again, false negatives can be avoided. In order to obtain good results, it is important to specify a large enough variance of the learned model. It is obviously problematic to estimate the variance from only very few training examples. Therefore, we have used the option to scale up the learned variance of descriptors and curve shape. Figures 3c,d show the difference between using unmodified and scaled values. We obtain much higher recognition rates without introducing false positives even if we scale up the standard deviation significantly, for both the trained shape model and the descriptors. Consequently, we use these settings for all other examples in our paper.

Running time: As most descriptor-based shape analysis techniques, our technique needs some time to precompute the descriptors. This has to be done only once per data set. Preprocessing times are in the range of a few minutes to up to one hour (Table 1). Please note that this is an embarrassingly parallel task that can probably be speed up significantly by an optimized implementation, but this is not the focus of our work. The interaction is much faster. Training a model is very fast but some auxiliary tasks such as estimating the global descriptor variance can take up to 5 seconds in our examples. Finding all line features is usually also interactive, in the range from below one to approximately 10 seconds. However, it is possible to create "broken" training models for which the algorithm takes a very long time (up to 20 minutes) to finish. This happens for example if curves are sketched on non-descriptive, planar regions where pruning cannot be effectively performed. For practical applications, these degenerate cases are not relevant as no reasonable matching can be expected. Nevertheless, we would like to include a user warning in such cases in future work.

Figure 4: *Different types of windows recongnized as one class in the experiment shown in Figure 3b.*

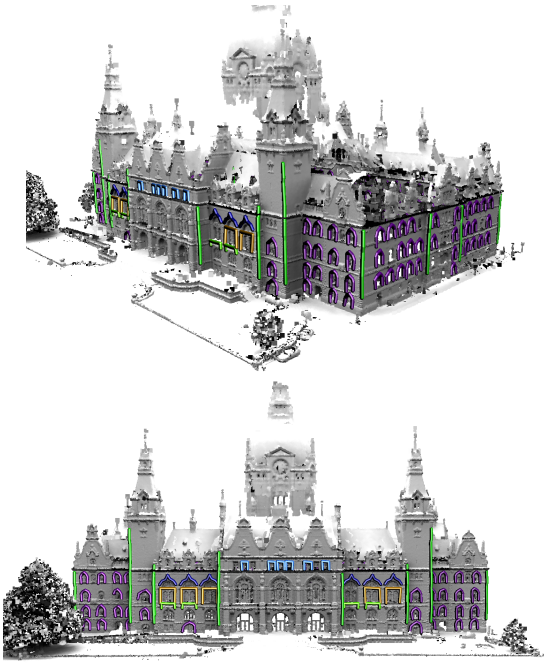

Figure 5: *Decomposition into several classes.*

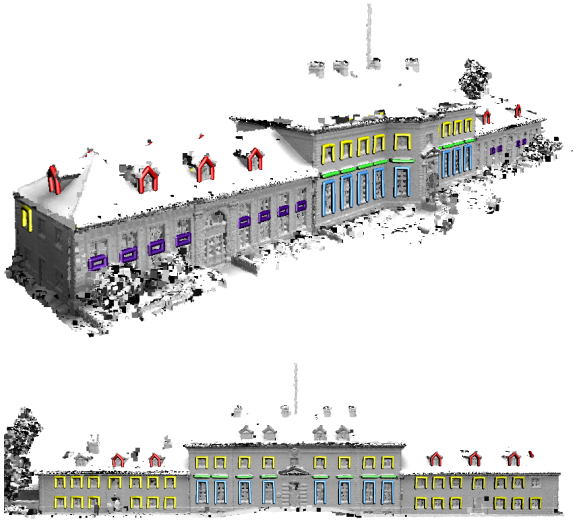

Figure 6: *5 different classes extarcted. Same global parameters for each class. Two training instances clicked per class*

7. Discussion

In any detection task, there is an inherent trade-off between generalization performance and precision. Our model aims at learning robustly from very few examples, which limits the precision for very large training sets. Figure 4 shows that,

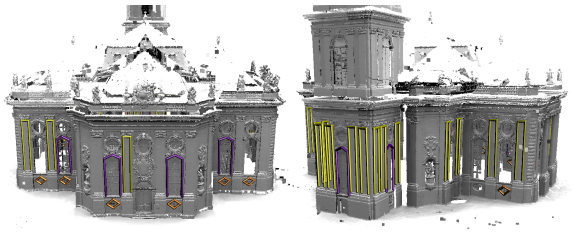

Figure 7: *Learning classes by using just one example each.*

nevertheless, quite some variance can be captured. And for very different geometry, the user of our system can anytime create multiple classes of feature lines with corresponding vertices to model complex mixtures. We also evaluate the ability to transfer learned features from one data set to another in Figure 8, depicting a row of different houses from one street in the Hannover data set: Only the small part of one house marked in blue is used for initial learning (4 windows). The model is stored and reused on the second, independent data set and a large fraction of windows in the other houses is detected. Of course, this is limited to rather similar geometry. It is impossible to infer very different geometry that is not within the span of the example space.

Figure 9 illustrates various quality levels of user input. For a single input curve, accuracy is not an issue and all matched curves will be detected with the same imperfect shape. Only when combining several curves in a single model, accuracy plays a role. Nonetheless, potential imprecision is implicitly handled by our model that already accounts for inaccurate surface descriptions. Further, as the feedback is almost immediate, it is easy to add only curves when needed or to see and correct the negative impact of an imprecise curve. Figure 9 illustrates the effect of different levels of noise in the specification of two example curves. It turned out that we had to deliberately click off target to obtain artifacts.

The bad-input example of Figure 9 also reflects that we do not explicitly impose closed curves. In principle, it is possible to formulate the belief propagation algorithm for closed curves, but the computation cost would increase by a polynomial factor. The choice of open curves allows us to perform

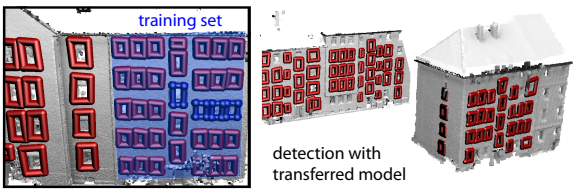

Figure 8: *Transferring learned models: Four curves were trained on the blue area, in red are detected windows for the same and a different data set.*

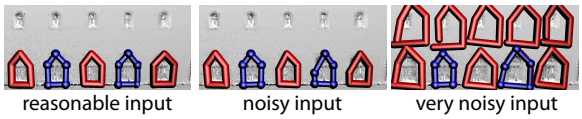

Figure 9: *Our method is robust to imprecisions in user input (top, middle), even for very noisy input, our solution still finds reasonable matchings (bottom).*

our very efficient search procedure. Most matched curves are almost closed because the descriptors snap them to the right locations during the search. We could easily add a culling step to reject curves that are not "closed enough", but, in practice, this was not necessary. In our examples, only for the unrealistically bad user input benefits from this.

Our model is not necessarily scale invariant: Only if the user provides examples with scale differences, the linear model will automatically span the whole subspace of scaled examples. However, our current descriptors are not scale-invariant, thereby limiting the scale range; a design choice to make the detection more reliable. We could achieve full scale-invariance using scale space descriptors such as SIFT [Low03, LG05]. Still, our solution is flexible enough to handle significant variations (Figure 10). By learning different scales its impact in the detection step is reduced (although we still use scale-dependent descriptors). The algorithm finds even instances that were not directly trained, but it fails for strong scale differences which is desired behavior.

Limitations and Future Work: Our user-guided feature detection scheme has currently some limitations. Most importantly, we can only find line features, we cannot find complex graphs with cycles. These have to be composed out of several, independently-trained and retrieved lines. In future work, we would like to examine matching more complex graphs by combining our approach with graph matching techniques such as [BBW*08,BBW*09b]. In this context, an extension to general, tree structured graphs as feature primitives would be possible. Furthermore, our current feature model assumes a fixed upward, mostly motivated by applications to architectural models and similar man-made structures. This is no principal limitation. However, a straightforward solution to this problem (such as evaluating the match for different base orientations covering 360° degrees) would increase the runtime. A more efficient strategy is again subject to future work. One more limitation we have observed in practice is that the model is not robust to missing data; if part of a feature maps to a hole in the surface it cannot be detected because no discretization points are available (using a robust ϕ model is therefore not sufficient). Here, a dynamic discretization of the domain, including potential holes, is needed. In addition to addressing these limitations, we would also like to examine more complex example applications, such user guided (partial) symmetrization in the spirit of Mitra et al. [MGP07], global matching of dissimilar shapes [ACP03, ZSCO*08], as well as shape modeling by docking corresponding parts, as introduced by Bokeloh et al. [BWS10], now replacing rigid docking with user-guided docking sites.

Figure 10: *Although our descriptors are not scale invariant to increase precision, our method is scale tolerant (green) when learned (red) and supports complex features (top: discontinuities, bottom: embossed shapes).*

8. Conclusions

We have presented a technique that learns line features from user input that is designed to work with few training examples. We believe that this weak form of user supervision is a powerful tool for shape matching problems in the future, in particular, if care is taken to minimize the necessary work for the user. Nonetheless, even a small user interaction makes matching significantly more robust. Although conceptually simple and easy to implement, our approach, hereby, obtains good results with few false positives and negatives. We can train both discriminative models (that find only rigidly similar instances), as well as broader classes of models (that relate to each other non-rigidly, but have a vaguely similar appearance). In our prior experience, similar results in accuracy and robustness have not been possible to achieve with unsupervised techniques. Furthermore, only setting up their parameters takes more time than our solution to draw a few sketches. Furthermore, a supervised approach like ours can learn broader classes of *semantically* similar objects, which is a very hard problem without user input. Consequently, user guidance could offer a viable way to higher-level "shape understanding", approaching semantic rather than purely geometric interpretations. We consider our algorithm a first step into this broad and less explored venue of geometry processing.

Acknowledgments

This work has been partially funded by the Cluster of Excellence "Multi-Modal Computing and Interaction". The authors wish to thank the anonymous reviewers for their valuable comments, Claus Brenner and the IKG Hanover for making their data available, and Tobias Leppla and the LKVK Saarland for the scan of the "Ludwigskirche".

References

[ACP03] ALLEN B., CURLESS B., POPOVIĆ Z.: The space of human body shapes: reconstruction and parameterization from

range scans. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM, pp. 587–594.

[ASP*04] ANGUELOV D., SRINIVASAN P., PANG H.-C., KOLLER D., THRUN S., DAVIS J.: The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *Proc. NIPS* (2004).

[ATCO*10] AU O. K.-C., TAI C.-L., COHEN-OR D., ZHENG Y., FU H.: Electors voting for fast automatic shape correspondence. In *Computer Graphics Forum (In Proc. of Eurographics 2010)* (2010), vol. 29.

[BBK06] BRONSTEIN A. M., BRONSTEIN M. M., KIMMEL R.: Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Science (PNAS) 103*, 5 (2006), 1168–1172.

[BBW*08] BERNER A., BOKELOH M., WAND M., SCHILLING A., SEIDEL H.-P.: A graph-based approach to symmetry detection. In *Proc. Symp. Point-Based Graphics 2008* (2008).

[BBW*09a] BERNER A., BOKELOH M., WAND M., SCHILLING A., SEIDEL H.-P.: *Generalized Intrinsic Symmetry Detection*. Tech. Rep. TR MPI-I-2009-4-005, MPI Informatik, 2009.

[BBW*09b] BOKELOH M., BERNER A., WAND M., SEIDEL H.-P., SCHILLING A.: Symmetry detection using line features. *Computer Graphics Forum (Eurographics 2009)* (2009).

[BM92] BESL P. J., MCKAY N.: A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell. 14*, 2 (1992).

[BR07] BROWN B., RUSINKIEWICZ S.: Global non-rigid alignment of 3-d scans. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26*, 3 (2007).

[BV99] BLANZ V., VETTER T.: A morphable model for the synthesis of 3d faces. In *Proc. SIGGRAPH* (1999), pp. 187–194.

[BWS10] BOKELOH M., WAND M., SEIDEL H.-P.: A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph. 29*, 4 (2010).

[CET01] COOTES T., EDWARDS G., TAYLOR C.: Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 6 (2001), 681–685.

[CM92] CHEN Y., MEDIONI G.: Object modelling by registration of multiple range images. *Image Vision Comput. 10*, 3 (1992), 145–155.

[CZ08] CHANG W., ZWICKER M.: Automatic registration for articulated shapes. *Computer Graphics Forum (Proc. of SGP) 27*, 5 (2008), 1459–1468.

[DG03] DASGUPTA S., GUPTA A.: An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures and Algorithms 22*, 1 (2003), 60–65.

[FH05] FELZENSZWALB P., HUTTENLOCHER D.: Pictorial structures for object recognition. *Intl. J. Computer Vision 61*, 1 (2005), 55–79.

[GMGP05] GELFAND N., MITRA N. J., GUIBAS L. J., POTTMANN H.: Robust global registration. In *Proc. Symp. Geometry Processing* (2005), pp. 197–206.

[HAWG08] HUANG Q.-X., ADAMS B., WICKE M., GUIBAS L. J.: Non-rigid registration under isometric deformations. *Computer Graphics Forum (Proc. SGP) 27*, 5 (2008), 1449 – 1457.

[HFG*06] HUANG Q.-X., FLÖRY S., GELFAND N., HOFER M., POTTMANN H.: Reassembling fractured objects by geometric matching. *ACM Trans. Graphics 25*, 3 (2006), 569–578.

[HPW05] HILDEBRANDT K., POLTHIER K., WARDETZKY M.: Smooth feature lines on surface meshes. In *Proc. Symp. Geometry Processing* (2005).

[HTB03] HÄEHNEL D., THRUN S., BURGARD W.: An extension of the icp algorithm for modeling nonrigid objects with mobile robots. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)* (2003), pp. 915–920.

[KHS10] KALOGERAKIS E., HERTZMANN A., SINGH K.: Learning 3d mesh segmentation and labeling. *ACM Trans. Graph. 29*, 3 (2010).

[LG05] LI X., GUSKOV I.: Multiscale features for approximate alignment of point-based surfaces. In *Symp. Geometry Processing* (2005), pp. 217–226.

[LLS04] LEIBE B., LEONARDIS A., SCHIELE B.: Combined object categorization and segmentation with an implicit shape model. In *Proc. ECCVŠ04 Workshop on Statistical Learning in Computer Vision* (2004).

[Low03] LOWE D.: Distinctive image features from scale-invariant keypoints. In *Int. J. Computer Vision* (2003), vol. 20, pp. 91–110.

[LTSW09] LASOWSKI R., TEVS A., SEIDEL H.-P., WAND M.: A probabilistic framework for partial intrinsic symmetries in geometric data. In *IEEE International Conference on Computer Vision (ICCV)* (September 2009).

[MGP06] MITRA N. J., GUIBAS L. J., PAULY M.: Partial and approximate symmetry detection for 3d geometry. *ACM Trans. Graph. 25*, 3 (2006), 560–568.

[MGP07] MITRA N. J., GUIBAS L., PAULY M.: Symmetrization. In *ACM Transactions on Graphics* (2007), vol. 26.

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-valley lines on meshes via implicit surface fitting. In *Proc. Siggraph* (2004), pp. 609–612.

[PG01] PAULY M., GROSS M.: Spectral processing of point-sampled geometry. In *Proc. Siggraph* (2001).

[PJC*10] PERINA A., JOJIC N., CASTELLANI U., CRISTANI M., MURINO V.: Object recognition with hierarchical stel models. In *Proc. Europ. Conf. Comp. Vision (ECCV)* (2010).

[PMW*08] PAULY M., MITRA N. J., WALLNER J., POTTMANN H., GUIBAS L.: Discovering structural regularity in 3D geometry. *ACM Transactions on Graphics 27*, 3 (2008).

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (2006).

[SSB05] SCHÖLKOPF B., STEINKE F., BLANZ V.: Object correspondence as a machine learning problem. In *Proc. Int'l Conf. on Machine Learning* (2005), pp. 777–784.

[SSB06] STEINKE F., SCHÖLKOPF B., BLANZ V.: Learning dense 3d correspondence. In *Advances in Neural Information Processing Systems (NIPS)* (2006), vol. 19, pp. 1313–1320.

[TBW*09] TEVES A., BOKELOH M., WAND M., SCHILLING A., SEIDEL H.-P.: Isometric registration of ambiguous and partial data. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)* (2009).

[VJ01] VIOLA P., JONES M.: Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)* (2001).

[YFW01] YEDIDIA J. S., FREEMAN W. T., WEISS Y.: Understanding belief propagation and its generalizations. In *IJCAI 2001 Distinguished Lecture track* (2001).

[ZSCO*08] ZHANG H., SHEFFER A., COHEN-OR D., ZHOU Q., VAN KAICK O., TAGLIASACCHI A.: Deformation-drive shape correspondence. *Computer Graphics Forum (SGP 2008) 27*, 5 (2008), 1431–1439.