Generalized Intrinsic Symmetry Detection

Alexander Berner, Martin Bokeloh,
Michael Wand, Andreas Schilling,
Hans-Peter Seidel

MPI–I–2009–4–005 August 2009

Authors' Addresses

Alexander Berner
Max-Planck-Institut für Informatik
Saarbrücken, Germany

Martin Bokeloh
Max-Planck-Institut für Informatik
Saarbrücken, Germany

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

Andreas Schilling
WSI / GRIS, Tübingen University
72076 Tübingen, Germany

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

Abstract

In this paper, we address the problem of detecting partial symmetries in 3D objects. In contrast to previous work, our algorithm is able to match deformed symmetric parts: We first develop an algorithm for the case of approximately isometric deformations, based on matching graphs of surface feature lines that are annotated with intrinsic geometric properties. The sensitivity to non-isometry is controlled by tolerance parameters for each such annotation. Using large tolerance values for some of these annotations and a robust matching of the graph topology yields a more general symmetry detection algorithm that can detect similarities in structures that have undergone strong deformations. This approach for the first time allows for detecting partial intrinsic as well as more general, non-isometric symmetries. We evaluate the recognition performance of our technique for a number synthetic and real-world scanner data sets.

Keywords

symmetry detection, shape matching, shape understanding

Contents

1 Introduction

Detection of symmetries in 3D objects has recently received a lot of attention in the geometry processing literature [Mitra *et al.*, 2006; Podolak *et al.*, 2006; Loy & Eklundh, 2006; Mitra *et al.*, 2007; Pauly *et al.*, 2008]. The goal of this line of work is to discover structural redundancies in objects in order to better "understand" their shape, which is helpful for several applications such as compression, shape completion, or providing high-level tools for shape editing. In a general sense, a symmetry of an object $\mathcal{S}$ can be defined in the following way: We are looking for a piece of geometry $\mathcal{U} \subseteq \mathcal{S}$ and a number of mapping functions $f^{(i)}$, that instantiate this piece in multiple instances, thereby matching the original geometry up to some error tolerance. Most of the previous work considers only reflections, rigid mappings and sometimes uniform scaling as mapping functions. However, many real-world objects show forms of structural redundancy that cannot be captured by such simple affine maps. For example, different leaves of a plant might clearly appear similar to a human observer, but often cannot be transformed into each other by a global affine mapping. Similar types of symmetries are observed for many objects such as different windows of a building or ornamental structures in man-made sculptures.

Figure 1.1: *A 3D scan of a deformed plasticine sculpture with three snail figures. Our algorithm detects symmetries by looking at constellations of surface feature lines. This can be used for detecting deformable symmetries. Left: input data, middle: feature lines overlay, right: detected symmetries.*

In this paper, we address the problem of *generalized symmetry detection*. This means, we want to find parts in object that might actually differ by a significant deformation but, to a human observer, would still appear to be of the same kind.

We look at two variants of this problem, which are of increasing difficulty: First, we look at isometric symmetries, where we can still assume that the deformations of the instances approximately preserves intrinsic distances. Afterwards, we extend our technique to handle cases of more general, non-isometric deformations. Our approach is based on feature matching. Effectively, feature extraction transforms the complex, under-constrained geometric matching problem into a simpler discrete graph matching problem. The features provide an abstraction that is much more invariant under typical differences of symmetric instances than the original geometry. Only in a second step, to obtain dense correspondences, we perform continuous geometry matching using the matched features as constrained, using a weak smoothness regularizer to interpolate in between. However, the symmetries are actually defined by matching recurring structures of salient features.

As feature representation, we employ points and lines of maximum curvature, corresponding to creases on the surface. This choice is motivated by human perception: for humans, crease lines are particularly salient cues for shape recognition [Kent *et al.*, 1996]. This fact is not surprising, as for many real-world objects, a few ridges and valleys already encode most of the geometric information [Ohtake *et al.*, 2004]. We annotate the network of feature curves with intrinsic properties of these curves, such as length, intrinsic angles, and intrinsic curvature, which are checked during graph matching. The resulting graph is invariant to isometric deformations of the surface by construction. In particular, this immediately yields an algorithm for partial intrinsic symmetry detection, which was not covered by previous work. This is the first main contribution of this paper. By relaxing the matching precision for these geometric attributes, i.e. permitting larger variations in lengths, curvature or angles, we obtain the generalized symmetry detection algorithm, that can still detect similar structures in objects that differ by significant deformations. To the best of our knowledge, such generalized symmetry detection has not been attempted before. Describing a first approach to deal with this problem is the second main contribution of this paper.

In order to evaluate the performance in practice, we apply the symmetry detection technique to a number of data sets with symmetric structures of varying similarity, ranging from rigid to not even isometrically similar. Even in the general case, we are able to identify many of the important symmetries that are apparent to a human observer fully automatically.

2 Related Work

The most successful group of symmetry detection techniques is based on transformation voting [Mitra *et al.*, 2006; Podolak *et al.*, 2006; Loy & Eklundh, 2006; Pauly *et al.*, 2008]. The key idea of these techniques is to parametrize the transformation functions with a small number of parameters (such as translation, rotation, scaling), find a set of candidate correspondences and vote for matching correspondences in a Hough space. Transformation voting techniques are currently probably the most frequently used, state of the art class of techniques. However, it is not obvious how to generalize the concept to more complex transformations that require many more parameters which might not be suitable for voting. For example, the parameters of a general free form deformation affect the deformed instances only locally so that voting in a global parameter space is not possible. A similar parametrization problem also applies to geometric hashing [Lamdan & Wolfson, 1988; Gal & Cohen-Or, 2006]. For other approaches, such as robust auto-alignment [Simari *et al.*, 2006], spherical harmonics analysis [Martinet *et al.*, 2006], or primitive fitting [Schnabel *et al.*, 2008], it is so far also unclear how to handle more general transformations.

Our algorithm is based on previous work that employs feature graph matching for symmetry detection [Martinet, 2007; Berner *et al.*, 2008; Bokeloh *et al.*, 2009]. The main challenge of feature-based methods is that the recognition performance depends decisively on that of the feature detection. [Berner *et al.*, 2008] use regions of minimum slippability and [Bokeloh *et al.*, 2009] employ crease lines as features, which yield particularly useful cues for shape matching. Our algorithm is also based on matching graphs of lines on surfaces. However, in contrast to previous work, our new approach is not limited to rigid symmetries. This is a non-trivial addition; the core step in [Bokeloh *et al.*, 2009] is a rigid alignment of local line pattern using iterative-closest-lines (ICL), which is not applicable in the case of deformed surfaces. In addition, we develop a robust randomized graph matching strategy that is insensitive to a small amount of topological noise in the

graph. This problem is of no concern in [Berner *et al.*, 2008], where absolute positions can be estimated by rigid transformation matrices.

Only few authors have so far addressed the problem of detecting symmetries under non-linear mappings: [Ovsjanikov *et al.*, 2008] present a technique for detecting global intrinsic symmetries of objects by analyzing eigenfunctions of the Laplace-Beltrami operator of the surface. This approach reveals in a very elegant way global symmetries based solely on intrinsic computations, and is therefore invariant to isometric deformations. However, due to the global nature of the eigenfunctions, it is unclear whether this approach can be used for partial symmetry detection, which is the aim of our paper. [Raviv *et al.*, 2007] detect symmetries by finding feature point sets that preserve geodesic distances followed by a numerical optimization of an isometric embedding by generalized multi-dimensional scaling (GMDS) [Bronstein *et al.*, 2006]. This approach does not take into account multiple simultaneous symmetries but rather aims at finding global symmetries in order to detect asymmetric irregularities. Both papers do not consider generalizations beyond isometric matching.

[Lasowski *et al.*, 2009] detect partial intrinsic symmetries in 3D geometry. Their algorithm is based on a probabilistic formulation of partial shape matching: based on a Markov random field model, they obtain a probability distribution over all possible intrinsic matches of a shape to itself, which reveals the symmetry structure of the object. Then they approximate marginals of this distribution using sumproduct loopy belief propagation. The approach scales quadratically in memory and cubically in computation time with model size. Therefore, despite its formal elegance, the approach is not (yet) very practical. Furthermore, it is unclear how to formulate a generalization beyond the isometric setting.

In shape retrieval, topological matching techniques have been used to recognize semantically similar shapes (see for example [Hilaga *et al.*, 2001]). Recently, [Zhang *et al.*, 2008] prosed a shape matching technique based on comparing graphs of extremity features and evaluating the induced deformation of a match, aiming at matching shapes such as humans or animals. The technique is able to compute matches between rather different objects, and is (to the best of our knowledge) currently the only technique that addresses this problem from a global optimization perspective. [Allen *et al.*, 2003] match different human body shapes using local optimization guided by (manually placed) markers, in a formulation very similar to our final continuous matching step. [Teves *et al.*, 2009] consider pairwise isometric matching of deformable surfaces using graph matching, focusing on robustness in the presence of holes and topological noise in the input surfaces. None of these techniques consider the case of detecting partial symmetries within objects.

3 Overview

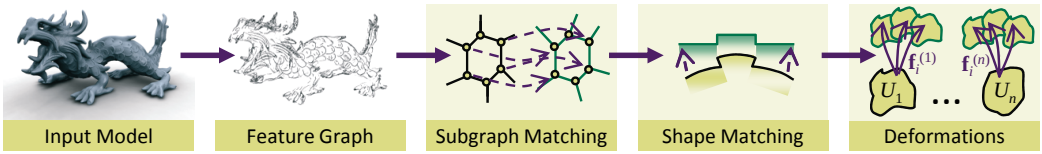

Figure 3.1: *Pipeline - we first detect a network of feature lines on the object surface. Next, we match the resulting graphs, which are annotated with intrinsic properties. By adjusting the admissible tolerance, we can continuously move from isometric to more general matching. The resulting discrete structures are used to initialize a numerical deformable shape matching step, which yields a set of deformation functions between symmetric instances as result.*

First, we give a brief overview of our technique: The processing pipeline is shown schematically in Figure 3.1. Our algorithm expects a point sampled representation of the manifold as input. This representation allows us to handle both 3D scanner data as well as triangle meshes, which can be easily sampled in a preprocessing step (we employ a uniform Poisson disc sampling in these cases). The first step of our processing pipeline is the detection of a network of feature lines on the object surface, as detailed in Section 4. This yields a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of surface lines, each of which is annotated by various geometric properties. In this graph, we then look for recurring subgraphs (Section 5). Afterwards, we employ local deformable shape matching (Section 6) to assign surface area to corresponding instances and compute dense correspondences, using the discrete matches to guide the alignment. We repeat this algorithm several times until no more significant symmetries are found. As final output, we obtain a *symmetry set* $\mathcal{R} = \{\mathcal{I}_1, ..., \mathcal{I}_n\}$ consisting of n *instance sets* $\mathcal{I}_i$. Each instance is encoded by a single representative "urshape" $\mathcal{U}_i$ and n_i deformation functions $f_i^{(j)} : \mathcal{U}_i \rightarrow \mathbb{R}^3, j = 1...n_i$ that deform the urshape to match the original geometry. These deformation functions establish dense correspondences between all surface pieces within an instance set.

4 Feature Detection

The goal of our paper is to detect symmetric objects in 3D shapes that appear to be of the same type to a human observer, although they cannot necessarily be mapped to each other by a simple transformation such as an affine map. In order to achieve this invariance, we first transform the input data into a discrete feature representation that captures the structure of an object in the topological arrangement of these discrete elements.

In the following, we describe two different approaches, an extrinsic and a purely intrinsic variant: The extrinsic feature detector (Subsection 4.1) is looking for crease lines of maximal principal curvature. This is probably the "canonical choice" for detecting feature lines: As detailed in [Ohtake *et al.*, 2004], crease lines contain the most important part of the overall geometric shape information and correspond to perceptually important shape cues. This is particularly important in the case of generalized symmetries: Here, the pattern of crease lines actually *defines* the notion of what is considered to be symmetric, in an attempt to coarsely mimic human perception. The theoretical downside of this approach is that principal curvatures are not isometrically invariant. In practice, this is usually no problem. Real-world variations in shapes are typically smooth such that small scale extrinsic properties are approximately preserved as well. However, for completeness, we also examine the case of a purely intrinsic feature definition in Subsection 4.2, which makes our approach completely agnostic to isometric deformations of the input.

4.1 Extrinsic Features

Our extrinsic feature detection framework was motivated by the "ridges and valley" extraction approach by [Ohtake *et al.*, 2004]. In principle, any feature detection algorithm [Gumhold *et al.*, 2001; Pauly *et al.*, 2003; Ohtake *et al.*, 2004;

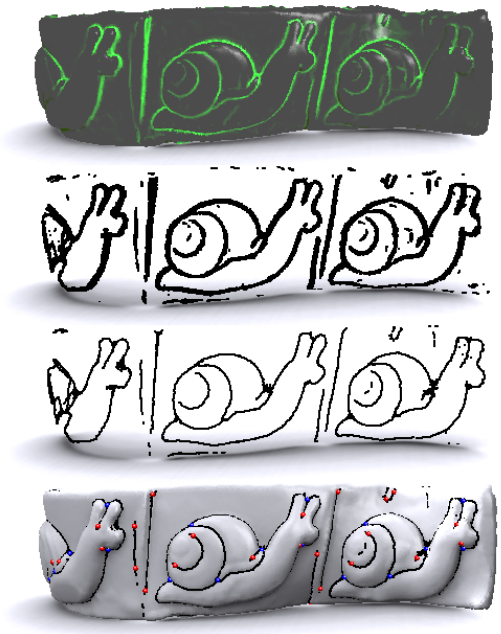

Figure 4.1: *Feature detection pipeline. From top to bottom: product of κ_1 and its second derivative, patches after thresholding, shrunken to lines, resulting feature graph.*

Hildebrandt *et al.*, 2005] could be used in conjunction with the rest of our symmetry detection pipeline. As feature detection is not the main focus of our paper, we opt for a simple and easy to implement technique. To get an intuition of typical results, Figure 4.1 shows an overview of the results obtained for the "snails" data set from Figure 1.1.

Feature Scale and Preprocessing: In the following, we describe how to extract features at a fixed scale ϵ_{scale}. Later, we will vary this scale in order to build a multi-resolution representation. We assume we are given a sampled representation of a surface $\mathcal{S}$. First, we resample the point set by deleting points that are closer to their nearest neighbors than $0.25\epsilon_{scale}$. This step makes sure that the costs for the subsequent computations do not become too large. Afterwards, we form a "topology graph" that connects each point to its 12 nearest neighbors (excluding edges longer than $3\epsilon_{scale}$, to make the construction robust to outliers). This graph encodes the apparent topology of the manifold and is used in all subsequent operations. Later in the algorithm, a subset of this graph will form the curve lines.

Curvature computation: Given a point $\mathbf{p}$, we consider the curvature tensor $\mathbf{C}(\mathbf{p})$ and its two eigenvalues κ_1 and κ_2, sorted by their absolute value. We estimate these quantities via a quadratic moving least squares (MLS) fit [Alexa *et al.*,

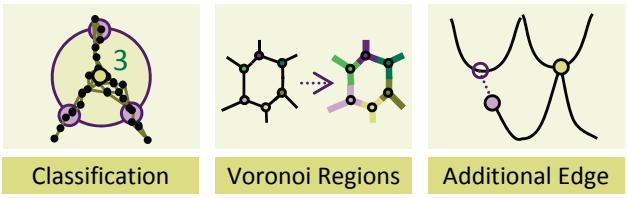

Figure 4.2: *Steps for building the feature graph. First, intersections and dead ends are detected as feature points. Second, Voronoi regions on the Feature lines are computed around all feature points in order to form a connectivity graph. Lastly, small gaps are closed to make the algorithm more robust.*

2003], using a Gaussian window of size ϵ_{scale}. In order to make the resulting values scale invariant, we normalize the local surface pieces by scaling by ϵ_{scale}^{-1} before MLS fitting. A crease line should maximize the larger of the two principal curvatures κ_1. In particular, this implies that the first derivative of κ_1 vanishes in the direction of $\mathbf{t}_1$ while the second derivative in this direction is non-zero. We compute this second derivatives by again fitting a second order MLS approximation to the values of κ_1 computed previously. We prefer this technique as it is more robust than attempting to directly fit a 4th order representation (the number of parameters to be estimated is much smaller, reducing overfitting).

Classification: We now classify candidate crease line points by looking at the product of the absolute value of the first principal curvature and its second derivative in the principal direction. We threshold this quantity to obtain a set of curve line candidates. In practice, the product of these two quantities is a quite robust measure for crease line detection: Even varying the threshold value by a factor of 10 or 100 does not change the solution drastically. In contrast, a direct thresholding of the curvature values is very sensitive to parameter setting; even changes by a factor of 1.1 can lead to substantially different results.

Feature line shrinking: As a result of the classification, we obtain narrow patches of crease line points, connected by the topology graph. In order to obtain smooth, linear curves, we apply a very simple filtering technique: For each point, we collect all neighboring points within a graph distance of no more than $12\epsilon_{scale}$ and move each point to the centroid of this set. The centroid is then projected back on the manifold using again a quadratic MLS surface fit. This smoothing is rather simplistic and tends to oversmooth highly curved lines; however, as it does so consistently, this is no problem in the context of our application.

Feature line graph: The next task is to convert the set of feature line points into a graph of lines and crossings of such lines. We employ a simple sphere crossing test: For each point on a feature line computed previously, we center a sphere

of radius $5\epsilon_{scale}$ around it and determine all pairs of connected curve points for which the connecting edge in the topology graph intersects with the sphere. If multiple such intersections occur within a distance of ϵ_{scale}, they are counted only once. This test yields a classification of curve points into line segments (2 intersections), dead ends (1 intersection) and crossings (3 or more intersections). For dead ends and crossings, each connected component of the same type is converted into a vertex in the graph. In order to form the graph edges, we simlutaneously run Dijkstra's algorithm starting from all vertices in order to compute Voronoi regions in the graph of line feature points. Whenever two Voronoi regions meet, their graph vertices are connected by an edge. The lengths of this edge is set to the according Dijkstra distance. In addition, we estimate the average integral geodesic curvature of the edge by fitting an MLS quadratic curve to each point along the Dijkstra path and averaging the values. We will later use this measure to qualitatively distinguish edges by their bending direction (left/right w.r.t. the travel direction), which is often at least qualitatively preserved even in general, non-isometric symmetries. We also compute the tangent vectors to the curve lines at the graph vertices, using finite differences. They will later be used to check intrinsic angles.

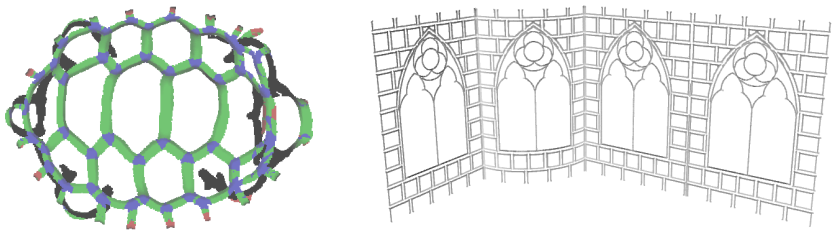

Figure 4.3: *Line features for the turtle and the window data set.*

Additional edges: In practice, the line feature detector is not always able to correctly detect intersections of line features, in particular if the height or depth of the crease lines is rather shallow. Even if the crease line actually vanishes before hitting another line, the human visual system would take the logical continuation into account and use this property for feature matching. Consequently, we detect "dead end" feature points that are close to other feature lines and extend their course in tangential direction, hitting the other line with a newly created intersection feature (see Figure 4.2, right).

4.2 Intrinsic Features

For purely intrinsic symmetry detection, we replace extrinsic pairs of principle curvatures by the intrinsic Gaussian curvature. We use the same MLS scheme for the computation, just multiplying the two principal values. We need to modify this scheme slightly to make it reliable in practice: Quadratic estimations with radially symmetric Gaussian weights can lead to a bias when being applied to a patch of finite size. Straight crease lines of complex cross section, but obviously with zero Gaussian curvature everywhere, may appear to have a phantom curvature in the direction of the crease line. This is because a spherical cut-out area might be best approximated in a least-squares sense by a quadratic polynomial that bends into two directions and sampling limitations prevent us from converging to an unbiased solution. We avoid this problem by using, in a second step, an unweighted square window with axes aligned to the previously computed directions of curvature.

The maxima of Gaussian curvature do not form line structures but point features. Therefore, we form our graph differently: First, we extract point features thresholding the Gaussian curvature and connecting the resulting peaks to their k nearest neighbors with geodesic paths on the surface (we use $k = 8$). We annotate each point feature with the sign of the Gaussian curvature (hyperbolic, spherical). Geodesics are again estimated using Dijkstra's algorithm on the topology graph. The output (graph of lines with crossings) is the same as of the first technique and use in the same way in the subsequent pipeline. However, this intrinsic graph provides less information than the extrinsic feature lines. In particular, the intrinsic curvature vanishes everywhere on the geodesic lines. However, we will only use it in the case of strict isometric matching, where we do have the stronger invariant of approximate distance preservation (which is not available in the general case of large non-isometric deformations). Therefore, this design trade-off is acceptable.

5 Graph Matching

The main component of our algorithm is a graph matching algorithm that detects similar feature constellations. The matching routine consists of two nested loops: The inner loop is a randomized greedy algorithm that generates random candidate solutions. Depending on the random choices made, different solutions (i.e., instance sets) of different quality will be output. The outer loop then iterates this routine several times in order to sample the solution space and only outputs the best instance set found. We can then apply this algorithm repeatedly, removing graph elements from already identified instance sets, to output all symmetries within an object.

5.1 Inner Loop

We assume we are given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of feature points and lines, corresponding to vertices and edges. Each such discrete element might be annotated by continuous geometric properties. Our task is to find a subset $\mathcal{U}_\mathcal{G} \subseteq \mathcal{G}$ of the feature graph and a set of discrete mapping functions $f_{\mathcal{G}^{(i)}}$ that map vertices and edges of this subset to corresponding subgraphs $\mathcal{U}_\mathcal{G}^{(i)}$ that have approximately the same structure. This means, that the graph topology itself should be similar and the annotations should be approximately preserved. By varying how strictly these geometric quantities are preserved, we can go from isometric to more general symmetry detection.

Start edge: Each greedy matching step starts by choosing a random edge from $\mathcal{E}$, which we will call *start edge* e_{start} in the following. The corresponding instance, which up to now contains only this single edge, will be called *start instance* $\mathcal{I}_{start}$. We now look for more instances that are similar to the start instances by comparing the start edge to all other edges e' in the graph. We denote this comparison function as probability $P(e, e')$ (we will give more details on the implementation

of this comparison function later, in paragraph "edge matching"). It is important to take into account that instances might be symmetric to themselves, resulting in overlapping but non-identical matches. Therefore, we distinguish directed edges (i, j) and (j, i). This corresponds to a rotation by $180°$. In order to also account for mirroring, we tag each instance with a "mirrored" bit, that indicates that the whole instance has changed its orientation (i.e., the cross product of a tangent frame flips its orientation with respect to the extrinsic surface normal). In the worst case, each start edge can appear three more times as starting matches, varying by rotation and mirroring.

Instance growing: So far, we have only a small graph of two vertices and one edge that corresponds to a a number of other such small graphs. Our task is thus to extend this with additional matches. This is done by first choosing a random vertex on the start instance. We then examine all outgoing edges $e_{candidate}$ that are not yet contained in the instance and evaluate how well they can be matched to the other instances. We then take the best such match, and reevaluate the other edges until no more matching edge is found. In order to evaluate an edge $e_{candidate}$, we go through all other instances and compute the best matching edge, i.e. an edge e_{match} that maximizes the matching probability $P(e_{candidate}, e_{match})$. Very low scores (below 0.2) are ignored. For each edge $e_{candidate}$, we compute the expected benefit, which is the sum of correctness probabilities $P(e_{candidate}, e_{match})$. Finally, we select the edge $e_{candidate}$ with the highest benefit and all its associated matches.

Selecting the solution: The iteration outlined above stops automatically once no more edges with sufficient matching score are found. After the iteration, we have a situation in which we know a number of correspondences between the start instance $\mathcal{U}_\mathcal{G}$ and several other subgraphs $\mathcal{U}_\mathcal{G}^{(i)}$. Each correspondence is typically partial, covering a subset of $\mathcal{U}_\mathcal{G}$ each. For now, this is no problem; the continuous shape matching step will later automatically determine a consistent common subset of geometry for each instance set. However, we need to remove spurious matches such as cases, in which only the start edge matched and nothing else. We do this by deleting all $\mathcal{U}_\mathcal{G}^{(i)}$ that are not supported by at least a minimum number k_{min} of vertex correspondences. The rational behind this strategy is that we need to gather some "evidence" that the match is reasonable. A very small number of correspondences might be correct plainly incidentally, while a larger number makes the match more believable. k_{min} is a user parameter that trades-off false positives versus missing some matches. If we have sufficiently strong geometric evidence, it is often sufficient to use rather small values such as $k \geq 4$.

Topological robustness: A feature graph for real-world data sets is usually far from perfect. Therefore, we have to deal with topological noise in the graph, which will lead to false positive intersections. This effect can occur for example if

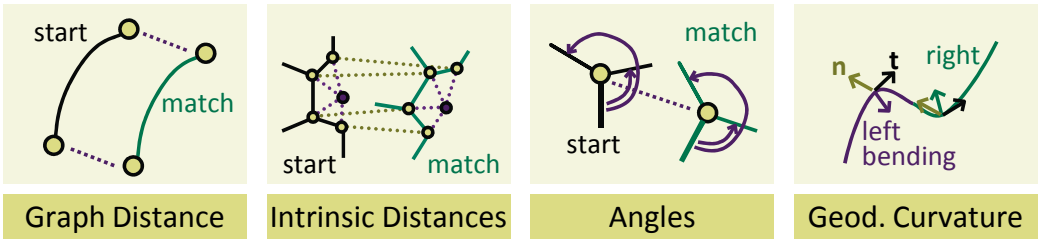

Figure 5.1: *We employ four geometric validation criteria plus a check of the graph structure (subgraph isomorphy). By setting the matching sensitivity for each of these criteria separately, we can generalize the algorithm to non-isometric matching. Typically, we obtain good results in practice by disregarding distances completely and using coarse penalties for angles and curvature.*

noise is converted into small line segments that form crossings that are not present in symmetric instances. Hence, we do not check only one outgoing edge but allow the algorithm to skip over intersection points. In case the correct matching edge is interrupted by a spurious intersections, the algorithm will try to go up to three points further, along outgoing lines with best matching tangential direction. This feature skipping is performed on both the start instance and matching instance side (conceptually, this is equivalent to just checking additional edges on both sides).

Edge matching: Finally, we need to design our edge matching function $P(e, e')$. We formulate it as a probability value between zero and one. The actual value will not only depend on the pair of edges themselves, but also on the two subgraphs that contain these edges and have already been matched previously. The matching probability is a product of five separate validation scores:

- **Graph structure:** If the two matching feature points do not have the same outdegree, they are unlikely to be a good match (not impossible, because there might be topological noise). We also check if a newly matched vertex is connected to a vertex that is already in the instance. In that case, the connecting edge must be present in both instances and connect to the same vertices (thus enforcing subgraph isomorphy). If any of this fails, we reduce the probability by a factor of 0.25.
- **Edge length:** The newly inserted edge should be of the same length. We assume Gaussian noise with a user specified standard deviation σ_{length} and output the corresponding probability density, normalized to the range [0..1].
- **Geodesic distances:** For this criterion, we compute the geodesic distances on the *original* surface $\mathcal{S}$ of the newly inserted feature point to all other feature in the start instance and the matched instance (again, using Dijk-

stra's algorithm). We again assume Gaussian noise with standard deviation σ_{geod}, and multiply the resulting probabilities for the deviation and output a normalized value.

- **Intrinsic angles:** We use an additional criterion where we check the angles of the tangents of the outgoing edges in the tangent plane with respect to the incoming edge. Again, strictness is controlled by a standard deviation parameter σ_{angle}. For the angle matching, we need to check the orientation bit of the instance and reverse the orientation of the angles in cases of reflected instances. The angle criterion is particularly useful in generalized matching scenarios. Angles are often preserved even if distances are not (in object classes such as windows, for example).
- **Geodesic curvature:** Finally, we also check the average geodesic curvature, parameterized by a standard deviation parameter σ_{curv}. Again, a qualitative match is useful for generalized symmetry detection (such as to distinguish between left-bending and right-bending curves). Again, we need to check the mirroring bit to reverse the bending direction in mirrored instances.

5.2 Outer Loop

The success of the inner loop depends on a number of random decisions. We therefore execute the procedure repeatedly to obtain several alternative instance sets, sampling the solution space[1]. From these, we chose only the best solution. In order to quantify what the best solution is, we look at the complexity of the result; more complex matches are less likely to be spurious: Given n instances with n_i graph elements each, we assign to this solution a score of: $\prod_{i=1}^{n} 1 - exp(-\lambda n_i)$ where λ is a user parameter in the range of $0.5...0.1$ (typically: 0.2). The ratio behind this heuristic is to compute the expected gain of the solution, where each instance is worth one unit and the probability of being a false positive drops exponentially with the number of graph elements used to validate the match (basically naively assuming independent and identically distributed probabilities for each validation step). This criterion is modeling the problem more closely than the ad-hoc quadratic instance weighting in [Bokeloh *et al.*, 2009].

[1]Our current formulation of the inner loop makes a number of deterministic decisions, which could also be randomized by making this choices at random with a probability reflecting the expected success. Such a strategy would then actually sample the complete solution space with high probability after a sufficiently long execution time. However, the simpler deterministic formulation works better in practice.

6 Deformable Shape Matching

Discrete shape matching gives us only sparse correspondences between symmetric instances. In a subsequent step, we propagate this information to the full geometry. As we have already solved the global combinatorial assignment problem, we can now use a locally convergent numerical shape matching technique. For strictly isometric symmetries, an isometric embedding such as the GMDS technique of [Bronstein *et al.*, 2006] would be a good choice (and straightforward to integrate into our framework). As we are also aiming at detecting more general symmetries, we opt for 3D thin-plate splines as introduced by [Allen *et al.*, 2003] for matching shapes with irregular variations. This regularizer does not preserve isometries exactly but, in practice, yields visually good solutions with the ability to handle general shape distortions. In addition, it gives a canonical extrapolation of the deformation field into the space surrounding the deformed object, which is useful for many applications.

We compute a deformation function $\mathbf{f} : \Omega \rightarrow \mathbb{R}^3$ that maps from a volumetric region Ω enclosing the undeformed shape into three space. The thin plate spline energy tries to keep local deformation gradients similar, i.e. minimizes the Hessian matrix of the deformation function [Allen *et al.*, 2003; Brown & Rusinkiewicz, 2007]. In addition, we add a constraint energy for shape matching, which leads to the following objective function:

$$E(f) = \int_{\Omega} \|\lambda_{reg}\mathbf{H_f}\|_F^2 + \sum_{i=1}^{n_{pos}} (\mathbf{f}(\mathbf{x_i}) - \mathbf{y}_i)^T \mathbf{M}_i (\mathbf{f}(\mathbf{x_i}) - \mathbf{y}_i)$$

$\mathbf{H_f}$ denotes the Hessian matrix of $\mathbf{f}$, λ_{reg} is a user parameter that controls the flexibility of the mapping, $\mathbf{x}_i$ and $\mathbf{x}_i'$ are positions in Ω at which the positions $\mathbf{y}_i$ and Jacobian matrices $\mathbf{Y}_i$ are prescribed, respectively. $\mathbf{M}_i$ are error quadrics that describe the weighting of the position constraints in all spatial directions.

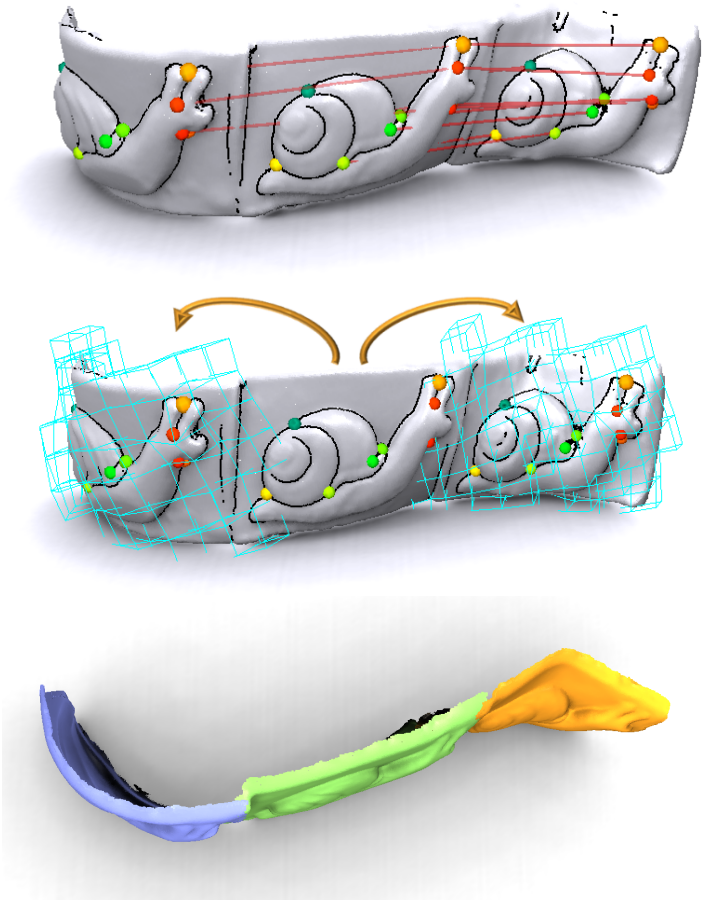

Figure 6.1: *More details on the matching from Figure 1.1. Left: Discrete vertex correspondences from the graph matching step. Middle: Continuous deformation using thin-plate splines. Right: A view from the top, to show the non-rigidity of the deformation.*

We solve the variational problem by discretizing $\mathbf{f}$ on regular grid of basis functions (as we will have a large number of constraints, this is more efficient than using globally supported radial basis functions of fundamental solutions). We use a sparse grid that has only entries near actual surface points: The grid is chosen such that the domain is overlapped by all possible basis functions with non-zero support and at least one more additional layer of grid cells to allow for extensions into the nearby volume. The derivatives are approximated as symmetric finite differences of adjacent grid cells. As basis functions, we use simple but efficient piecewise linear functions during the iterations of the deformable alignment and a more costly smooth interpolation with radial Wendland functions in the final iteration where all correspondences have already been established. The discretization

leads to a linear system of equations that we solve using a standard conjugate gradients algorithm.

Using this machinery, we can implement a deformable ICP algorithm: We start by constructing the undeformed parametrization domain (urshape) by cutting out the bounding cube of the matched features out of the start instance point set. Next, we setup position constraints that map intersection points to the positions on the matching shape (with identity quadrics $\mathbf{M}_i$). We also map lines to lines using a proportional mapping with respect to arc length as initial guess. With this initial guess, a first alignment is computed. Afterwards, we compute for each deformed point the closest point on the target surface and setup a point-to-plane constraint that attracts the deformed point to the closest target in normal direction. This is implemented by setting the error quadric to a rank one tensor given by the outer product of the normal to the target surface with itself. For each constraint, we also add a copy displaced by the surface normal scaled by the grid size. This helps in keeping the volumetric deformation problem well constrained. We solve the resulting linear system and iterate the deformable ICP until convergence.

We use this to align all instances to the starting instance, which becomes our urshape. After completion, we use a simple region growing algorithm to cut out the symmetric geometry: We start at a random point close to the start edge and in a breadth-first marching algorithm simultaneously extend the covered region. This leads to a growing front of constant geodesic distance that eventually either collides with other fronts or reaches a region with large point-to-plane residuals where the growing stops because of mismatching geometry. To make this outlier robust, we do not stop if less than 20% of the instances are affected by such a collision or a geometry mismatch. If growing reaches the boundary of the initially cut out domain before being stopped, the domain is extended and the iteration continued.

7 Implementation and Results

We have implemented our symmetry detection algorithm in C++ to evaluate the performance in practice. The results were obtained on PCs equipped with 2.4GHz Intel Core 2 Duo processors.

Deformed windows: Our first test data set is a synthetic geometry of four gothic windows, one of which is folded onto a cylinder, yielding a mapping very close to isometry (Figure 7.4). The discrete graph matching identifies the symmetric windows correctly, including the reflective symmetry. For this rather simple case, we obtain almost perfect correspondences. In a second step, we make this problem more challenging by applying a global non-isometric deformation to the scene (see Figure 7.5). In this case, the generalized symmetry detection algorithm is still able to find the four instances of the window, including reflective symmetries that lead to 8 overall instances. There are small artifacts in the corners of the instances; in this region, the deformable ICP was unable to extrapolate the correspondences precisely.

Plasticine snails: In order to have a real-world data set with approximately isometric symmetries, we have modeled and scanned a physical 3D model ourselves. The model consists of a sheet of plasticine in which three figures of snails have been impressed using a plastic cast. Subsequently, the plasticine sheet was bent to an S-shaped cross section. The data set has been acquired with a Minolta laser triangulation scanner, aligned using deformable ICP and postprocessed manually, which includes outlier removal, hole filling and MLS filtering for noise suppression. The result is shown in Figures 1.1 and 6.1. The resulting data set still has several imperfections: for example, the antennae of the head of the snails are flattened out in two instances as well as the back of the leftmost snail (due to falling on the floor before scanning). In addition, the surface shows scratches from sculpting that create outlier lines. These issues have intentionally not been fixed to obtain a realistic test case. After adjusting some parameters (see discussion below), we obtain a good matching results.

Figure 7.1: *Comparison of different symmetry detection algorithms. The top row shows an example of a relief that with rigid symmetris (left), near-isometrically deformed symmetries (middle) and non-isometrically deformed symmetries (right). The second row shows the resulting symmetric instances detected using the technique by [Bokeloh et al. 2009]. The third row shows our results, for the purely isometric pipeline (Gaussian curvature features, strict preservation of intrinsic distances). The last row shows the results for the generalized symmetry detection scheme (relaxed geometric matching, extrinsic crease features). As expected, the rigid technique fails on deformed examples. Our technique can also handle the case of more general deformations when parameterized accordingly.*

Comparison of matching strategies: In Figure 7.1, we compare different matching strategies; the top row shows a relief with three instances each that are rigidly displaced on the left, approximately isometrically deformed in the middle and strongly distorted (using manually placed constraints on a thin-plate spline energy) on the right. We then first run the most closely related technique of [Bokeloh *et al.*, 2009], which is state-of-the-art for feature based symmetry detection. As expected, the technique is not able, even with optimized parameters, to identify the deformed instances. However, the rigid instances are detected reliably. Next, we employ our isometric matching pipeline using intrinsic features based on Gaussian curvature (all other examples use the extrinsic features) and strict threshold on all intrinsic geometric quantities. As a result, the isometrically deformed parts are recognized reliably, but the technique fails for the more general cases. These are recognized by running our approach in "generalized settings" (using extrinsic crease curve detection). Please note that the recognition quality for the simpler cases does not suffer visibly from the generalization here.

Scanned turtle: Our next example data set is the scanned turtle figurine from [Rusinkiewicz *et al.*, 2002]. The main symmetry pattern is in the scutes (bumps) on the turtles back. This is a general, non-isometric symmetry. Our algorithm

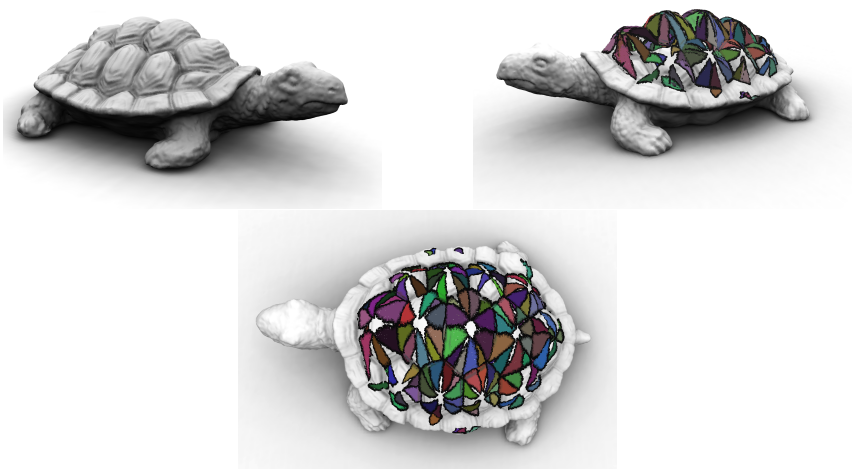

Figure 7.2: *General symmetries detected on the back of the turtle figurine. The differently colored pieces are all instances of a single symmetry; they show the result of applying the deformations $f^{(i)}$ to the urshape. Due to the rotational and reflective symmetry, we obtain (up to) 12 pieces per scutes. A small area in the middle of each scute remains empty because we do not explicitly constrain the instances to form periodic patterns. Data from [Rusinkiewicz et al. 2002].*

retrieves about 170 instances that cover a large portion of the symmetries present (see Figure 7.2). Each scute is 6-way rotationally symemtric to itself and each such piece has again a reflective symmetry. Our algorithm recognizes this fact and outputs up to 12 pieces as result. We currently do not attempt to detect regular patterns; therefore, the symmetries are not closing perfectly but a small area in the middle remains without correspondences.

More general symmetry examples: We have tested our algorithm on two more example data sets that contain generalized symmetries. The first is a piece of a castle data set, also used (for comparison) in [Bokeloh *et al.*, 2009], see Figure 7.3. The rigid matching technique recognizes just a global reflective symmetry for this data set, because the three gates are of different cross section. The generalized technique actually recognizes that these three instances are structurally similar, including the reflective symmetry within each gate. A second example is a piece of the back of the Stanford dragon, which features structurally similar scales that are easily recognized by a human observer. However, the actual geometry is varying drastically so that this similarity is very hard to detect for an algorithmic feature detector. Consequently, our technique is not able to detect all scales but at least detects the majority of them and cuts out reasonable instances automatically by the deformable alignment (see Figure 7.6). We have also tried to run the algorithm on

the full dragon data set; however, this failed due to the high memory requirments, exceeding the 2GB limitation of our 32bit implementation. Therefore, we provide the results for the left half of the complete data set as reference, which we would expect to generalize for the full data set (see Figure 7.7. In this case, we again are able to match only about half of symmetries in the scales (which are automatically recognized as the most salient symmetric structure); however, given the difficulty of the matching problem, this is a reasonable result.

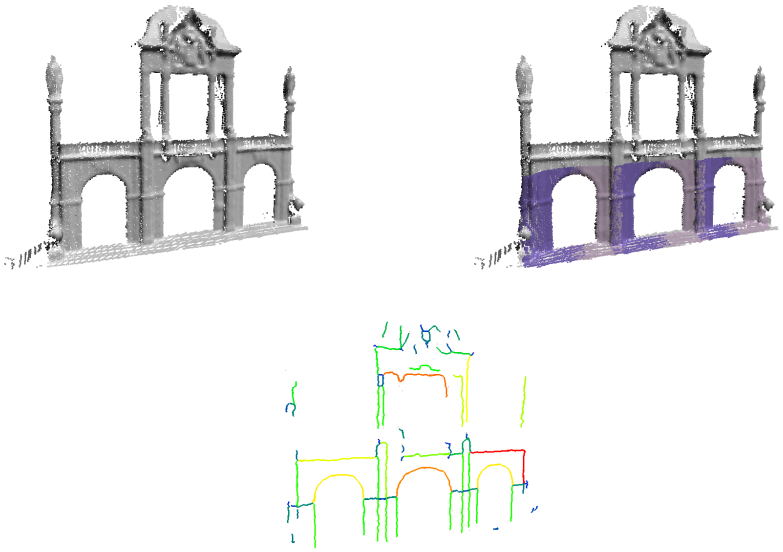

Figure 7.3: *A simple example for using general symmetries on a scanned architectural data set: One of the three gates is of different proportions; our technique recognizes the symmetry, including reflective in each piece symmetry. The rigid technique of [Bokeloh et al. 2009] only recognizes a single global reflective symmetry. The right image shows the detected feature graph (edge lengths are color coded).*

Running time: The most expensive parts of our computation pipeline are the feature detection step (in particular, the MLS computations, which performs many nearest neighbor queries) and deformable alignment. Both require computation times in the range of a quarter of an hour. In comparison, graph matching is inexpensive, with running times of less than a minute for all our data sets. Most of this time is spend on building the the feature graph and computing geodesic distances. The core graph matching routine always completes within a few seconds.

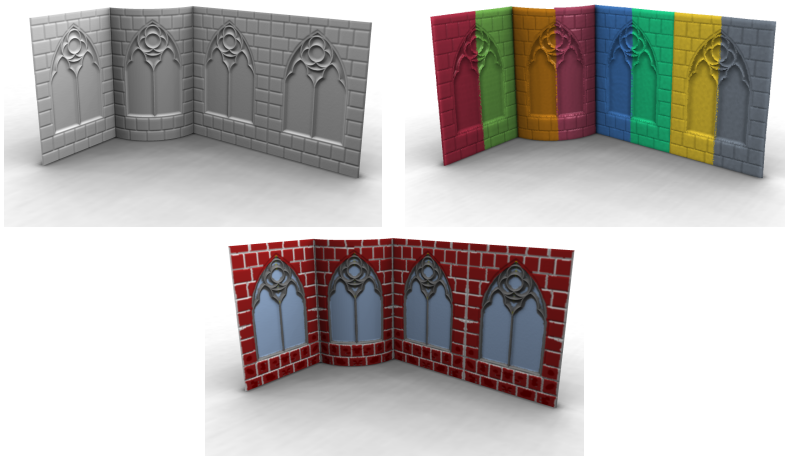

Figure 7.4: *A synthetic example for partial isometric symmetries; left: original geometry; middle: detected instances after region growing; right: painting on one instance and transferring to all others.*

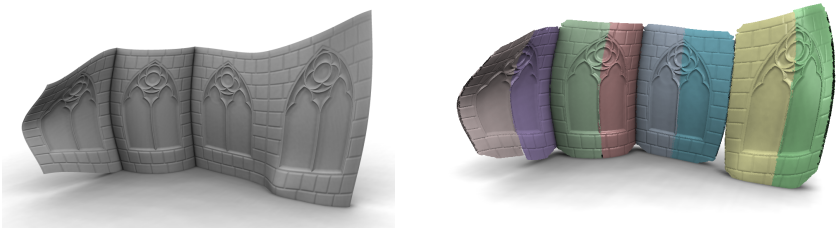

Figure 7.5: *Another synthetic example for partial isometric symmetries. Left: Original deformed geometry. Right: Detected instances after region growing; the image shows the result of deforming the urshape (half a window) by the computed continuous deformation function.*

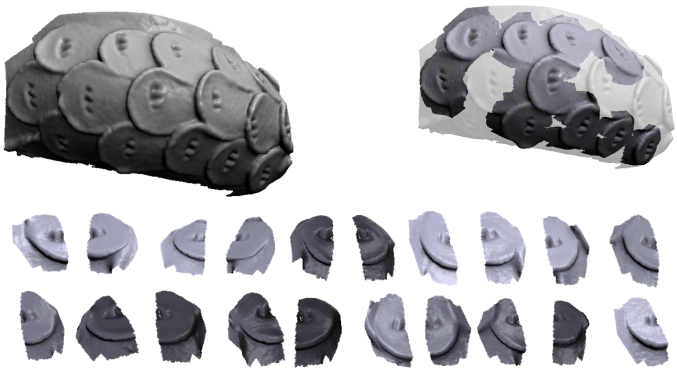

Figure 7.6: *Results for a small piece of the Stanford Asia Dragon data set. Our algorithm correctly identifies more than half of the repeating scales. Due to the strong geometric and even topological variation, this is a very hard problem. The images on the right show the deformed urshapes, which are very close in shape to the original geometry and match salient feature lines correctly.*

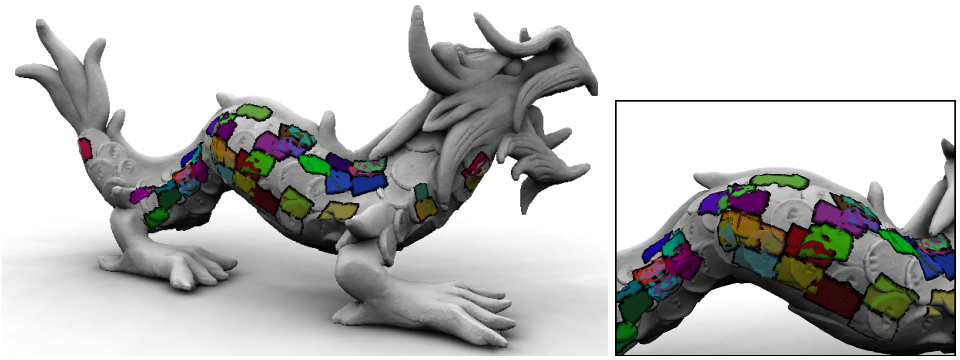

Figure 7.7: *Matching results for a larger piece of the Stanford Asia Dragon data set. Our technique is still able to recognize a substantial subset of the scales on the back, which are automatically identified as most salient symmetric structure.*

7.1 Discussion and Limitations

Our algorithm is able to identify isometric symmetries and establish dense correspondences. For more general symmetries, it is still able to retrieve matches a human observer would expect. However, we do not obtain perfect results, due to the difficulty of the general matching problem.

As most feature-based algorithms, the technique is parameter dependent. The most important parameter is the scale parameter ϵ_{scale}. Feature detection is rather parameter insensitive; however, handling difficult inputs such as the the damaged parts of the snails require some fine tuning of threshold, smoothing radius and curve shrinking parameters.

By design, the important parameters of the graph matching algorithm are the four standard deviations $\sigma_{length}, \sigma_{geod}, \sigma_{angle}, \sigma_{curv}$. These parameters can be used to adapt the matching algorithm to the specific situation. For general matching, filtering by angles and geodesic curvature (with increased standard deviation, though) might still yield good results while preservation of lengths is only an option for near-isometric cases. The only additional parameter we currently need to set is the minimum instance size to filter out small outlier instances; this value is not determined automatically with the current strategy. For region growing and deformable matching, we use a fixed parameter set for all cases.

Besides parameter dependence, our algorithm is mostly limited by the type of features we build our discrete structure on. The choice of high curvature lines works well in many different data sets, but currently fails if a surface contains only closed curves without intersections. However, refining the feature detector with additional cues (such as curve corners, slippable regions, geometric primitives etc.) is straightforward.

8 Conclusion and Future Work

We have presented a new algorithm to detect general symmetries in 3D shapes. Our algorithm can handle partial isometric symmetries, which has not been demonstrated before, as well as more general symmetries where the preservation of intrinsic geometric quantities can be relaxed. We consider this a first step towards solving a very difficult problem that has not been addressed before. Using our approach, we are able to detect complex symmetry patterns in shapes that previously could not be detected automatically.

In future work, we would like to address some of the limitations of our current system. In particular, the feature detection technique could be improved in terms of parameter dependence and generality. It would also be interesting to extend the notion of generalized symmetries further to objects that cannot be mapped one-to-one, but show similar structure, such as pieces consisting of a varying number of similar subpieces. Finally, it would be very interesting to examine the usage of general symmetry information to devise user friendly shape editing tools that partially automate common editing tasks.

References

Alexa, M., Beht, J., Cohen-Or, D., Fleishman, S., Levin, D., & Silva, C.T. 2003. Computing and rendering point set surfaces. *IEEE Trans. Visualization and Comp. Graphics*, **9**(1), 315.

Allen, Brett, Curless, Brian, & Popović, Zoran. 2003. The space of human body shapes: reconstruction and parameterization from range scans. *Pages 587–594 of: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*. New York, NY, USA: ACM.

Amenta, N., & Kil, Y. J. 2004. Defining point-set surfaces. *ACM Trans. Graph.*, **23**(3), 264–270.

Anguelov, Dragomir, Srinivasan, Praveen, Pang, Hoi-Cheung, Koller, Daphne, Thrun, Sebastian, & Davis, James. 2004. The Correlated Correspondence Algorithm for Unsupervised Registration of Nonrigid Surfaces. *In: Proc. NIPS*.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., & Wu, A. Y. 1998. An Optimal Algorithm for Approximate Nearest Neighbor Searching. *Journal of the ACM*, **45**, 891–923.

Bendels, G. H., Degener, P., Wahl, R., Körtgen, M., & Klein, R. 2004. Image-Based Registration of 3D-Range Data Using Feature Surface Elements. *Pages 115–124 of: Proc. VAST*.

Berner, Alexander, Bokeloh, Martin, Wand, Michael, Schilling, Andreas, & Seidel, Hans-Peter. 2008. A Graph-Based Approach to Symmetry Detection. *In: Proc. Symp. Point-Based Graphics 2008*.

Besl, P. J., & Mckay, N.D. 1992. A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, **14**(2).

Biber, P., & Straer, W. 2003a. The Normal Distributions Transform: A New

Approach to Laser Scan Matching. *In: IEEE/RJS International Conference on Intelligent Robots and Systems*.

Biber, P., & Straer, W. 2003b. Solving the Correspondence Problem by Finding Unique Features. *In: 16th International Conference on Vision Interface*.

Bokeloh, Martin, Berner, Alexander, Wand, Michael, Seidel, Hans-Peter, & Schilling, Andreas. 2009. Symmetry Detection Using Line Features. *Computer Graphics Forum (Eurographics 2009)*.

Bronstein, Alexander M., Bronstein, Michael M., & Kimmel, Ron. 2006. Generalized multidimensional scaling: A framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Science (PNAS)*, **103**(5), 1168–1172.

Brown, Benedict, & Rusinkiewicz, Szymon. 2007. Global Non-Rigid Alignment of 3-D Scans. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, **26**(3).

Castro, E. De, & Morandi, C. 1987. Registration of translated and rotated images using finite Fourier transforms. *IEEE Trans. Pattern Anal. Mach. Intell.*, **9**(5), 700–703.

Chen, Y., & Medioni, G. 1992. Object modelling by registration of multiple range images. *Image Vision Comput.*, **10**(3), 145–155.

Comaniciu, Dorin, & Meer, Peter. 2002. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, **24**(5), 603–619.

Curless, B., & Levoy, M. 1996. A Volumetric Method for Building Complex Models from Range Images. *Computer Graphics*, **30**, 303–312.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Pages 269–271 of: Numerische Mathematik*, vol. 1. Mathematisch Centrum, Amsterdam, The Netherlands.

Duda, R. O., Hart, P. E., & Stork, D. G. 2000. *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Felzenszwalb, P., & Huttenlocher, D.P. 2005. Pictorial Structures for Object Recognition. *Intl. J. Computer Vision*, **61**(1), 55–79.

Fischler, Martin A., & Bolles, Robert C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, **24**(6).

Frigo, M., & Johnson, S. G. 1998 (May). FFTW: An adaptive software architecture for the FFT. *Pages 1381–1384 of: Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, vol. 3.

Frome, Andrea, Huber, Daniel, Kolluri, Ravi, Bulow, Thomas, & Malik, Jitendra. 2004. Recognizing Objects in Range Data Using Regional Point Descriptors. *In: Proc. Europ. Conf. Comp. Vision (ECCV)*.

Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin, D., & Jacobs, D. 2003. A search engine for 3D models. *ACM Trans. Graph.*, **22**(1), 83–105.

Gal, R., & Cohen-Or, D. 2006. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph.*, **25**(1), 130–150.

Gal, Ran, Shamir, Ariel, Hassner, Tal, Pauly, Mark, & Cohen-Or, Daniel. 2007. Surface Reconstruction using Local Shape Priors. *In: Proc. Symp. Geometry Processing*.

Gatzke, T., Grimm, C., Garland, M., & Zelinka, S. 2005. Curvature Maps for Local Shape Comparison. *Page 244256 of: Proc. Shape Modeling International*.

Gelfand, N., Mitra, N. J., Guibas, L. J., & Pottmann, H. 2005. Robust Global Registration. *Pages 197–206 of: Proc. Symp. Geometry Processing*.

Gelfand, Natasha, & Guibas, Leonidas J. 2004. Shape segmentation using local slippage analysis. *Pages 214–223 of: Proc. Symp. Geometry processing*.

Gelfand, Natasha, Ikemoto, Leslie, Rusinkiewicz, Szymon, & Levoy, Marc. 2003. Geometrically Stable Sampling for the ICP Algorithm. *In: Proc. Int. Conf. 3D Digital Imaging and Modeling*.

Gumhold, S., Wang, X., & MacLeod, R. 2001. Feature Extraction from Point Clouds. *In: Proc. Meshing Roundtable*.

Harris, C., & Stephens, M. 1988. A Combined Corner and Edge Detection. *Pages 147–151 of: Proc. 4th Alvey Vision Conference*.

Hilaga, Masaki, Shinagawa, Yoshihisa, Kohmura, Taku, & Kunii, Tosiyasu L. 2001. Topology matching for fully automatic similarity estimation of 3D shapes. *Pages 203–212 of: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM.

Hildebrandt, Klaus, Polthier, Konrad, & Wardetzky, Max. 2005. Smooth Feature Lines on Surface Meshes. *In: Proc. Symp. Geometry Processing*.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., & Stuetzle, W. 1992. Surface reconstruction from unorganized points. *In: Proc. Siggraph 1992*.

Huang, Qi-Xing, Flöry, Simon, Gelfand, Natasha, Hofer, Michael, & Pottmann, Helmut. 2006. Reassembling Fractured Objects by Geometric Matching. *ACM Trans. Graphics*, **25**(3), 569–578.

Jenke, P., Wand, M., Bokeloh, M., Schilling, A., & Straßer, W. 2006. Bayesian Point Cloud Reconstruction. *In: Proc. EUROGRAPHICS '06*.

Kadir, T., & Brady, M. 2001. Saliency, Scale and Image Description. *Int. J. Computer Vision*, **V45**(2), 83–105.

Kazhdan, M., Chazelle, B., Dobkin, D., Funkhouser, T., & Rusinkiewicz, S. 2003a. A Reflective Symmetry Descriptor for 3D Models. *Algorithmica*, **38**(1), 201–225.

Kazhdan, M., Funkhouser, T., & Rusinkiewicz, S. 2003b (June). Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors. *In: Proc. Symp. Geometry Processing*.

Kent, J. T., Mardia, K. V., West, J. M., & Jt, Leeds Ls. 1996. Ridge curves and shape analysis. *Pages 43–52 of: In The British Machine Vision Conference*.

Laboratory, Stanford Graphics. *Stanford Scanning Repository*. http://graphics.stanford.edu/data/3Dscanrep/.

Lamdan, Y., & Wolfson, H. J. 1988. Geometric hashing: A general and efficient model-based recognition scheme. *In: Proc. Int. Conf. Computer Vision*.

Lasowski, Ruxandra, Tevs, Art, Seidel, Hans-Peter, & Wand, Michael. 2009 (September). A Probabilistic Framework for Partial Intrinsic Symmetries in Geometric Data. *In: IEEE International Conference on Computer Vision (ICCV)*.

Li, X., & Guskov, I. 2005. Multiscale Features for Approximate Alignment of Point-based Surfaces. *Pages 217–226 of: Symp. Geometry Processing*.

Li, Xinju, Guskov, Igor, & Barhak, Jacob. 2006. Robust Alignment of Multi-view Range Data to CAD Model. *In: Proc. Shape Modeling and Applications*.

Lowe, D. 2003. Distinctive image features from scale-invariant keypoints. *Pages 91–110 of: Int. J. Computer Vision*, vol. 20.

Loy, G., & Eklundh, J.O. 2006. Detecting Symmetry and Symmetric Constellations of Features. *Pages 508–521 of: Proc. Europ. Conf. Computer Vision*.

Martinet, Aurélien, Soler, Cyril, Holzschuch, Nicolas, & Sillion, François. 2006. Accurate Detection of Symmetries in 3D Shapes. *ACM Trans. on Graphics*, **25**(2), 439 – 464.

Martinet, Aurlien. 2007. *Structuring 3D Geometry based on Symmetry and Instancing Information*. Ph.D. thesis, INP Grenoble.

Mitra, N. J., Gelfand, N., Pottmann, H., & Guibas, L. 2004. Registration of Point Cloud Data from a Geometric Optimization Perspective. *In: Symp. Geometry Processing*.

Mitra, N. J., Guibas, L., & Pauly, M. 2007. Symmetrization. *In: ACM Transactions on Graphics*, vol. 26.

Mitra, Niloy J., Guibas, Leonidas J., & Pauly, Mark. 2006. Partial and approximate symmetry detection for 3D geometry. *ACM Trans. Graph.*, **25**(3), 560–568.

Nilsson, P. E. Hart N. J., & Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems, Science, and Cybernetics*, **SSC-4**(2), 100–107.

Novotni, M., & Klein, R. 2004. Shape retrieval using 3D Zernike descriptors. *Computer Aided Design*, **36**(11), 1047–1062.

Novotni, Marcin, & Klein, Reinhard. 2003. 3D zernike descriptors for content based shape retrieval. *Pages 216–225 of: Proc. Solid Modeling and Applications*.

Ohtake, Yutaka, Belyaev, Alexander, & Seidel, Hans-Peter. 2004. Ridge-valley lines on meshes via implicit surface fitting. *Pages 609–612 of: Proc. Siggraph*.

Ovsjanikov, Maks, Sun, Jian, & Guibas, Leonidas. 2008. Global Intrinsic Symmetries of Shapes. *In: Eurographics Symposium on Geometry Processing (SGP)*.

Pauly, M., Mitra, N. J., Wallner, J., Pottmann, H., & Guibas, L. 2008. Discovering Structural Regularity in 3D Geometry. *ACM Transactions on Graphics*, **27**(3).

Pauly, Mark, Keiser, Richard, & Gross, Markus. 2003. Multi-Scale Feature Extraction on Point-Sampled Models. *In: Proc. Eurographics*.

Pauly, Mark, Mitra, Niloy, Giesen, Joachim, Gross, Markus, & Guibas, Leonidas J. 2005. Example-Based 3D Scan Completion. *In: Proc. Symp. Geometry Processing*.

Podolak, Joshua, Shilane, Philip, Golovinskiy, Aleksey, Rusinkiewicz, Szymon, & Funkhouser, Thomas. 2006. A Planar-Reflective Symmetry Transform for 3D Shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, **25**(3).

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press.

Raviv, Daniel, Bronstein, Alexander M., Bronstein, Michael M., & Kimmel, Ron. 2007. Symmetries of non-rigid shapes. *In: Proc. Workshop on Non-rigid Registration and Tracking through Learning*.

Rusinkiewicz, S., & Levoy, M. 2001. Efficient Variants of the ICP Algorithm. *Pages 145–152 of: Proc. 3rd Intl. Conf. 3D Digital Imaging and Modeling*.

Rusinkiewicz, Szymon, Hall-Holt, Olaf, & Levoy, Marc. 2002. Real-time 3D model acquisition. *ACM Trans. Graph.*, **21**(3), 438–446.

Schnabel, R., Wessel, R., Wahl, R., & Klein, R. 2008. Shape Recognition in 3D Point-Clouds. *In: Proc. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision*.

Schnabel, R., Degener, P., & Klein, R. 2009 (March). *Completion and Reconstruction with Primitive Shapes*. Computer Graphics Forum (Eurographics 2009).

Simari, Patricio, Kalogerakis, Evangelos, & Singh, Karan. 2006. Folding meshes: hierarchical mesh segmentation based on planar symmetry. *Pages 111–119 of: Proc. Symp. Geometry Processing*.

Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., & Seidel, H.-P. 2004. Laplacian surface editing. *Pages 175–184 of: SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. New York, NY, USA: ACM.

Teves, Art, Bokeloh, Martin, Wand, Michael, Schilling, Andreas, & Seidel, Hans-Peter. 2009. Isometric Registration of Ambiguous and Partial Data. *In: Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*.

Thrun, Sebastian, & Wegbreit, Ben. 2005. Shape from Symmetry. *In: Proc. Int. Conf. Computer Vision*.

Turk, G., & Levoy, M. 1994. Zippered polygon meshes from range images. *Pages 311–318 of: SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*.

Wand, Michael, & Straßer, Wolfgang. 2002. Multi-Resolution Rendering of Complex Animated Scenes. *Computer Graphics Forum (Eurographics 2002)*, **21**(3). Eurographics 2002.

Wand, Michael, Jenke, Philipp, Huang, Qi-Xing, Bokeloh, Martin, Guibas, Leonidas, & Schilling, Andreas. 2007. Reconstruction of Deforming Geometry from Time-Varying Point Clouds. *In: Proc. Symp. Geometry Processing*.

Yamany, Sameh M., & Farag, Aly A. 2002. Surfacing Signatures: An Orientation Independent Free-Form Surface Representation Scheme for the Purpose of Objects Registration and Matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, **24**(8), 1105–1120.

Zelinka, S., & Garland, M. 2004. Similarity-based surface modelling using geodesic fans. *Pages 204–213 of: Proc. Symp. Geometry processing*.

Zhang, Hao, Sheffer, Alla, Cohen-Or, Daniel, Zhou, Qingnan, van Kaick, Oliver, & Tagliasacchi, Andrea. 2008. Deformation-Drive Shape Correspondence. *Computer Graphics Forum (SGP 2008)*, **27**(5), 1431–1439.