Vision, Modeling, and Visualization (2011)
Peter Eisert, Konrad Polthier, and Joachim Hornegger (Eds.)

Partial Symmetry Detection in Volume Data

J. Kerber[1,2] M.Wand[1,2] J. Krüger[3,4] H.-P. Seidel[1,2]

[1] MPI Informatik Saarbrücken, [2] Saarland University,
[3] IVDA Group & IVCI, Germany, [4] SCI Utah, USA

Abstract

In this paper, we present an algorithm for detecting partial Euclidean symmetries in volume data. Our algorithm finds subsets in voxel data that map to each other approximately under translations, rotations, and reflections. We implement the search for partial symmetries efficiently and robustly using a feature-based approach: We first reduce the volume to salient line features and then create transformation candidates from matching only local configurations of these line networks. Afterwards, only a shortlist of transformation candidates need to be verified using expensive dense volume matching. We apply our technique on both synthetic test scenes as well as real CT scans and show that we can recover a large amount of partial symmetries for complexly structured volume data sets.

Keywords: symmetry detection, volume matching, information visualization

Categories and Subject Descriptors (according to ACM CCS):
I.4.7 [Computing Methodologies]: Image Processing and Computer Vision—Feature Measurement
I.4.8 [Computing Methodologies]: Image Processing and Computer Vision—Scene Analysis
I.5.4 [Computing Methodologies]: Pattern Recognition—Applications

1. Introduction

Gaining insight into the structure of volume data sets is a prime challenge in scientific visualization. Because of the dense 3D structure with occlusions, it is often hard to understand important structural properties of volume data for a human observer. This motivates algorithms that extract structural properties of such data sets fully automatically and and display them to the user. Obviously, this is not just useful in visualization of single data sets but also for an automated analysis or batch screening of large quantities of data.

In this paper, we look at one specific approach for extracting structural properties from 3D volume data: Partial symmetry detection. The goal is to find parts within the data that approximately map to other parts of the data under a Euclidean transformation (i.e., rotations, translations, reflections, and any combinations of these).

Partial symmetry detection has recently gained a lot of interest in the geometry processing community [MGP06, BBK06, GCO06, LE06, PSG*06, SKS06, OSG08, PMW*08, BBW*08, BBW*09] and has been proven to be an invaluable tool for a number of applications, ranging from denoising and compression to complex shape understanding algorithms that for example infer building plans from symmetry information.

Despite this success in processing 2-manifold data, very little work has been done in applying symmetry detection to 3D volume data. Our paper aims at filling this gap and proposes an efficient and robust computational framework for computing partial symmetries in 3D volume data.

Detecting such symmetries naively requires an exhaustive search over all Euclidean motions that map the volume (partially) to itself. As this group of transformations has 6 degrees of freedom, a well-sampled exhaustive search would require a computation time of $O(n^9)$ for an $O(n^3)$ volume data set, which is prohibitively expensive. Therefore, the main task of a practical symmetry detection algorithm is to reduce the amount of comparisons while maintaining the important self-matches.

As detailed in the next section, a number of approaches have been proposed that address this problem for 3D surfaces, but only very little work has been done in the domain of volume data. We use some ideas from [BBW*09], who

use line features as intermediate representation for efficient matching. This approach has the advantage of condensing the information to a small amount of characteristic features early in the matching pipeline, which is particular useful for volume datasets: On the one hand, this reduces the amount of data to be processed and on the other hand, this abstraction step helps in making the algorithm more robust: By reducing the matching to important feature structures, objects with complex structure can be matched automatically. We discuss how to perform feature extraction and matching effectively and robustly in 3D volume data and propose and evaluate a complete symmetry detection pipeline for the volumetric case.

We apply our algorithm to a number of test scenes, including two real-world CT scans of complexly structured objects.

2. Related Work

Symmetry Detection on Surfaces and Images: A considerable amount of work has been done in finding global and partial symmetries in surface data. Early work focused on exact, combinatorial algorithms [AMWW88]. More recently, algorithms have been explored that can handle inaccuracies and approximate matching. A large class of algorithms is based on transformation voting [MGP06, GCO06, LE06, PSG*06, PMW*08]: The algorithm builds a space of all transformation and votes for likely transformations by counting or clustering pairs of potentially matching surface elements. An alternative approach is numerical optimization, where shapes are initialized in a relative pose and an energy function is minimized that finds an overlap [SKS06, BBK06]. A third class of algorithms is based on matching graphs of features on surfaces by a geometric transformations [BBW*08, BBW*09]. Finally, global symmetries of shapes can also be computed by spectral moments [MSHS06]. Using Eigenfunctions of the Laplace-Beltrami operator of the surface, bending invariant symmetries [OSG08] can be found.

Applications: Symmetry detection has been used for a number of interesting applications. Thrun et al. [TW05] use symmetry to infer missing geometry from partial information. Another canonical application is the improvement of data quality [PMW*08, BBW*09] by using statistics of ensembles of symmetric objects. More recently, symmetry detection has been proven useful as an elementary building block in higher level shape understanding algorithms, such as structure aware shape editing [GSMCO09] and inverse procedural modeling [SBM*10, BWS10].

Volume analysis: Only very few methods have been proposed for detecting symmetries in volume data sets. The only technique we are aware of that specifically targets 3D voxel data is the work of Hong and Shen [HS08]. They apply the reflective symmetry transform algorithm of Podolak et al. [PSG*06] to volume data. In contrast to our approach, this method can only handle global and reflective symmetries. The methods based on spherical harmonics analysis [MSHS06] are inherently volumetric some of the other detection principles could be generalized to volume data as well, but we are not aware of concrete approaches.

While symmetry detection has not yet gained much attention in volume visualization, there is a large body of literature of techniques that extract structure from point clouds for a semi-automatic analysis. We give a brief overview, which is by no means exhaustive.

Many approaches have been studied for feature detection in volumes: Wu et al. [WCHK03] extract salient points in a multi-scale framework. Tzeng and Ma [TM05] extract and visualize 4D features within time dependent volume data via machine learning methods. Vijay et al. [VNP*05] present a topological approach for simplifying continuous functions defined on volumetric domains based on a small set of atomic functions. Caban et al. [CJR07] focusing on time-varying volume data with a texture-based technique tracks various features individually and then uses the tracked objects to better visualize structural changes. Hadwiger et al. [HLRS*08] present a method optimized for industrial CT (e.g. machine parts, such as our examples). The idea is based on a region growing preprocessing step that stores for each voxel a feature size curve that is then during rendering used in combination with transfer functions to interactively explore the features in the volume. Schulze et al. [SMB10] introduce probabilistic boosting trees with partial cascading and classifier sorting for general feature detection in volumes but focus mostly on medical data.

There are also several domain specific techniques: Laney et al. extract topological features to study Rayleigh-Taylor instabilities [LBM*06]. Grottel et al. [GRVE07] detect condensation clusters in Molecular Dynamics simulation data.

3. Symmetry Detection Algorithm

We first give a brief overview of our approach. The method proceeds in three major steps:

Feature Detection: Our algorithm uses constellations of line features as primary representation for the analysis and data matching. This reduces the amount of data to be processed and is the key to efficient computations. We employ line features because they preserve much more structural information than traditional key points [Low03], without excessive costs. We start with feature preserving filtering in order to suppress noise in the input volume. Then, every voxel is described by the differential properties of its local neighborhood and we extract a skeleton of salient structures. After shrinking and pruning the selected areas, locations of similar properties are grouped together to crease lines. The detection of locations where different crease lines intersect concludes the first part. Details are given in Section 3.1.

Sparse matching: Given a set of intersection points, we compare all contained junctions by determining the transformation between each pair. The transformations are rated according to the quality of the matching for adjacent line fragments. The best transformations, in the sense of most overlap found, are kept for the final step, as a transformation *candidate shortlist*. Details of the sparse feature matching step are discussed in Section 3.2.

Dense matching: For each transformation on the final candidate shortlist, we compute a separate *symmetry volume* by applying the transformation to each voxel of the input volume and comparing its value to the intensity at the destination position. We accordingly colorize the patches where the transformation has mapped to similar entries. Furthermore we visualize rotation axes and reflectance planes to better illustrate the applied transformation. Dense Matching is described in Section 3.3.

Visualization: We discuss briefly how we visualize the results in Section 3.4.

Postprocessing: Because of noise and quantization errors, the computed symmetry volumes usually still show some remaining noise and outliers. Therefore, we perform a postprocessing step to address this issue. See Section 3.5.

3.1. Feature Detection

Let $V : [1,n_x] \times [1,n_y] \times [1,n_z] \to [0,1]$ be a volumetric function that maps from a cuboid in $\mathbb{R}^3$ to real numbers between zero and one. We assume that we are given a discrete representation that stores values only at integer positions, i.e., on a regular grid. We assume that it is possible to distinguish between an actual object and the background by either thresholding very small values, or by a user defined additional binary mask.

Since the preservation of sharp edges and transitions between different materials is mandatory for the following steps, we opted for a bilateral filtering as an initial denoising [TM98]. It is known to ensure a smoothing of local high frequencies without blurring large intensity differences. This preprocessing helps to increase the robustness of the later mapping step for which the presence of noise could lead to outliers. We denote the filtered Volume by $\mathcal{V}$.

The remainder of this section summarizes the employed crease line extraction algorithm. It is based on the method recently developed by [KBW*10] where we refer for more details.

In the beginning, a quadratic three dimensional surface is fitted to the local neighborhood of each voxel. Deviations between the approximated surface and the actual behavior of the sub-volume are taken into account by a bilateral reweighting and a new refined surface is computed. This is repeated iteratively until convergence. The resulting surface approximates the local conditions best in a least square sense.

Given such a surface, we can describe each voxel by a structure tensor in the shape of an ellipsoid. Let $(\vec{N},\vec{K}_1,\vec{K}_2)$ be the orthonormal axes of this tensor pointing into the orientation of the surface normal and its principal curvature directions. Further, let the triple (g,k_1,k_2) be the magnitudes of the gradient and the bendings of the surface. In contrast to faces, where only a high gradient is present, edges exhibit significant values of the maximal principal curvature as well. The vector $\vec{K}_2$ in every voxel characterizes the direction in which an edge is locally propagating. All voxels are ranked according to the product $g \cdot k_1$ which is proportional to the area of the ellipse spanned by the two vectors. We pick the best percentage μ of all voxels for further processing.

Usually, the chosen voxels are situated in a certain radius around an actual edge. This spatial extension is shrunken by applying a connectivity preserving bilateral mean-shift filter. In this case, deviations in the orientation of $\vec{K}_2$ are weighted and the shift in each step is restricted to a plane perpendicular to this orientation. By this we ensure the preservation of connectivity in the filtered outcome. At the end of this step all points belonging to the same edge, collapsed to a thin line in space.

After that, all points which exhibit a similar orientation are clustered by a region growing approach. Each such cluster is henceforth denote as a **line feature** described by a common orientation and a set of points in space. Straight edges contain only one line feature, whereas curved or circular edges are described by multiple smaller elements of similar orientation.

Whenever, points of feature lines with deviating orientations converge close enough, we regard it as an intersection. These constellations are denoted as *bases*. Each base consists of a triple formed by a virtual intersection point in space and two line features (p,l_1,l_2). Note that also each counterpart (p,l_2,l_1) is detected. The set of the extracted bases $\mathcal{B}$ serves as input for the next step in the pipeline.

Figure 1 (a) shows the extracted feature lines for a binary cube. They are color-coded by their orientation and the junction points of the detected bases are indicated by small spheres. Exactly six distinct bases are located in each corner of the cube because three pairs of lines meet there.

3.2. Sparse Matching

Due to the abstraction which was achieved by the previous step, the size of the search universe has decreased drastically. The number of elements is small enough such that we now can afford an exhaustive search over all pairs of bases which requires $O(|\mathcal{B}|^2)$ comparisons. To do so, we proceed with an iterative closest line approach (ICL) which was initially used for detecting similar structures in point clouds [BBW*09]. The quality of this match is determined by taking into account the similarity of the adjacent line segments.

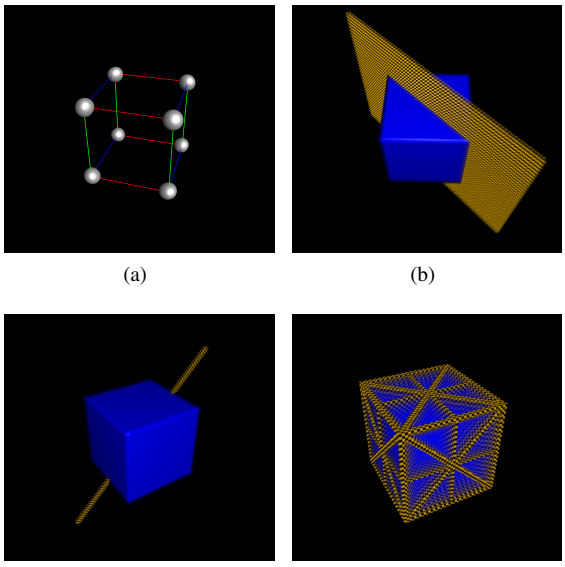


Figure 1: *Detected line features and intersections (a) two selected transformations (b+c) and a visual combination of detected symmetries (d) for a binary cube.*

Line feature matching starts by estimating an initial transformation from the matched base pair. Given this transformation, the adjacency graph of neighboring lines is traversed in a region growing approach: Whenever we visit a new line segment in the source instance, we search the target instance for a nearby line. We first locate the closest line sample and then compute the point-to-line distance by projecting the distance vector into the space orthogonal to the tangential direction of the line. We then evaluate a matching score: For each distance d_i of line sample from the source connecting to the closest target, we sum up the value of $\exp(-d_i^2/2\sigma^2)$. In other words, we assume a radial function around the lines within which geometry is considered to be matching. The radius of the region, given by σ, is set according to the expected errors in the match (due to noise, modeling errors, quantization problems, etc). Region growing is stopped if all neighboring line segments fall below a small threshold. The overall matching score accumulated so far is stored and used later in order to assess the match quality.

Refinement: The initial transformation estimates are obviously rather noisy. Therefore, we optimize the alignment iteratively: After each region growing step, we compute the point to line distance of all source points with respect to their closest target lines, weighted by the exponential kernels around each line. We then compute the transformation that minimizes the sum of these distance values (see [BBW*09] for details). After refinement, we continue region growing, and iterate (we currently use 5 iterations).

Removing doublets: The algorithm presented so far is likely to report one and the same match multiple times, obtained from different bases that yield a similar transformation, after iterative refinement. Let $\mathcal{T}$ denote the set of transformations obtained from the previous step. We prune $\mathcal{T}$ by eliminating similar transformations and only keep the highest ranked representative according to the matching score of the ICL algorithm. After that we proceed with eliminating inverse transformations; they are redundant as they are easily obtained from the originals by swapping source and target labels. Since each transformation is represented by a 4×4-Matrix, this is done by thresholding the Frobenius-norm, after normalizing the translation part to [0,1].

$$||T_i - T_j||_F < \varepsilon \quad (1)$$

$$||(T_i \cdot T_j) - ID_{4\times 4}||_F < \varepsilon \quad (2)$$

Given the example of the cube, this means we have 48×47 valid transformations in the beginning. After solving all double occurrences the size drops to 47 and further to 33 entries when the inverse transformations are eliminated. Figure 1 (b) and (c) show two of the remaining transformations: one rotation and one reflection. They are depicted by the corresponding axis and plane respectively.

3.3. Dense Matching

Now we can apply the remaining transformations to the volume itself and check for similarities in the overlapping areas. This means that for each element $t_i \in \mathcal{T}$ we create a new symmetry volume that separately shows the corresponding effect. According to the current transformation, each voxel in $\mathcal{V}$ is mapped to a destination. If this destination is located in the background or outside of the defined volume, then there is nothing to do. Otherwise we compare the entries at the source and target positions. If the intensity deviation is below a certain threshold both voxels are marked accordingly.

$$|\mathcal{V}(v) - \mathcal{V}(t_i(v))| < \tau_I \quad (3)$$

Here, τ_I denotes the allowed tolerance w.r.t. intensity deviations.

In general the target will not exactly match a voxel center. Experiments have shown that a trilinear interpolation is time consuming and causes wrong classifications at sharp boundaries and transition areas because of the continuous fall off. To overcome this, we opted for a rounding to the next discrete grid cell. This solution is much simpler, faster and the remaining problems can be solved as described below.

In the case of a rotation or a reflection there will be voxels which are either mapped to themselves or they have at least one direct neighbor for which both map to each other.

$$v_j = t_i(v_j) \quad (4)$$

$$(t_i(v_j) = v_k) \wedge (t_i(v_k) = v_j) \wedge (|v_i - v_j| < 2) \quad (5)$$

We specially mark these voxels because they belong to the corresponding axes or planes which help us visually describe the transformation. If a translation is involved we will not find any voxels which fulfill the above mentioned criterion.

3.4. Visualization

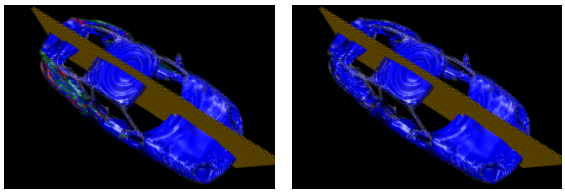
For the coloring of the symmetry volume we assign red to all voxels which are marked as a source and green to those which describe a target location. Areas where both labels overlap are colored in blue. To provide an additional orientation, the remaining foreground voxels of the initial volume are shown in a semi transparent light gray. We use a standard rendering approach of volumetric raycasting with alpha blending and Phong-like shading.

If voxels are marked as belonging to a symmetry plane or axis, we indicate this by a semi transparent checkerboard pattern with black and orange fields. This highlights the invariant voxels and assists a viewer in understanding the performed transformation. The absent of such voxels that are invariant under the symmetry transformation indicates a translational component in the mapping.

If we had not eliminated the inverse matrices in the matching step we would get additional volumes for which the red and green color would only be interchanged.

3.5. Post Processing

The continuous nature of the transformations and a potentially coarse resolution of the volume might cause outliers in form of discretization artifacts. This means that areas where source and target should have met, are only assigned to either one of them. As it can be seen in Figure 2(a), where red and green areas are present in the back of the car. These discontinuities are very thin, often not larger than one voxel. Therefore we apply an additional morphological erosion as a post processing (only for red and green) whereas the deleted voxels are reset to semi transparent light gray. The tidy result can be seen in Figure 2(b).



(a) (b)

Figure 2: *A reflective symmetry for a toy car model before and after post processing.*

4. Results

Figure 1 shows a toy example, a synthetic cube that has global symmetries, forming its octahedral symmetry group: The cube entirely maps to itself under multiples of 90° rotations around axes and axis aligned reflections. This shows as blue voxels in the visualization (no erosion was applied). In Figure 1 (d) we show the planes of all reflective transformations in one picture (the rotation axes are a part of the planes). Given this visualization the viewer can see that a cube consists of 48 equally shaped tetrahedra which start in each corner and point towards the center.

In Figure 3 we analyzed a synthetic binary writing and show 7 selected symmetries. The accompanying video contains additional matches and also demonstrates that for this inherently two dimensional model, rotations and reflections lead to the same results. Moreover it also contains point symmetries.

The engine block data set consists of many rotational symmetries, and the upper and lower part correspond in large areas as well. In addition, we also discover symmetries between disjoint volumetric areas (see Figure 4 and the accompanying video). For some of these symmetries, a human observer would probably not notice intuitively that these parts are actually related. We would therefore argue that this is a good example to demonstrate the use of our technique: We can robustly discover structural redundancy in complex and noisy data sets. Furthermore, we obtain insights into the structure of the data that is not apparent in a traditional volume rendering.

Finally we show the detected symmetries for a mechanical part in Figure 5. The plane and the coloring in the second image show that almost the entire volume is reflectively symmetric. The gray areas indicate where the geometry starts to differ. A rotation around the center axis is shown, followed by the detection of a pair of cylinders. This data set has less redundancy than the engine block but we still discover some main partial symmetries fully automatically. Again, this is real-world data from a CT scanner, showing that the abstraction into line features and our subsequent matching and filtering pipeline is effective in handling such data robustly.

4.1. Discussion

One limitation of the presented approach is that in the feature detection step, the quality of voxel is compared globally. In areas of low intensity the magnitudes of gradients and curvatures are naturally small. Hence it is likely that potentially less important features in other regions are preferred because they are ranked higher. Boundaries of cylinders and sphere are ranked high as well and might be detected (false positive) at the cost of proper edges.

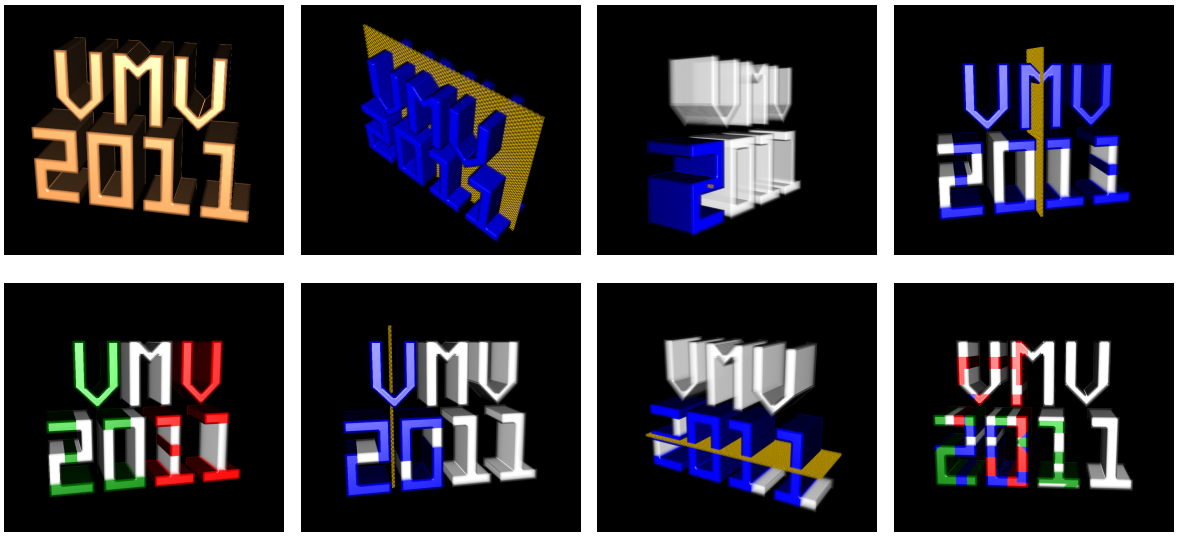


Figure 3: *A synthetic dataset and seven detected pairwise symmetries. Each figure shows a single symmetry transformation. Red: source volume, green: target volume, blue: both (overlap). Checked organge: self-mappings (i.e., planes of reflection or axes of rotation. The absense of self-mapping points indicate a translational component.*

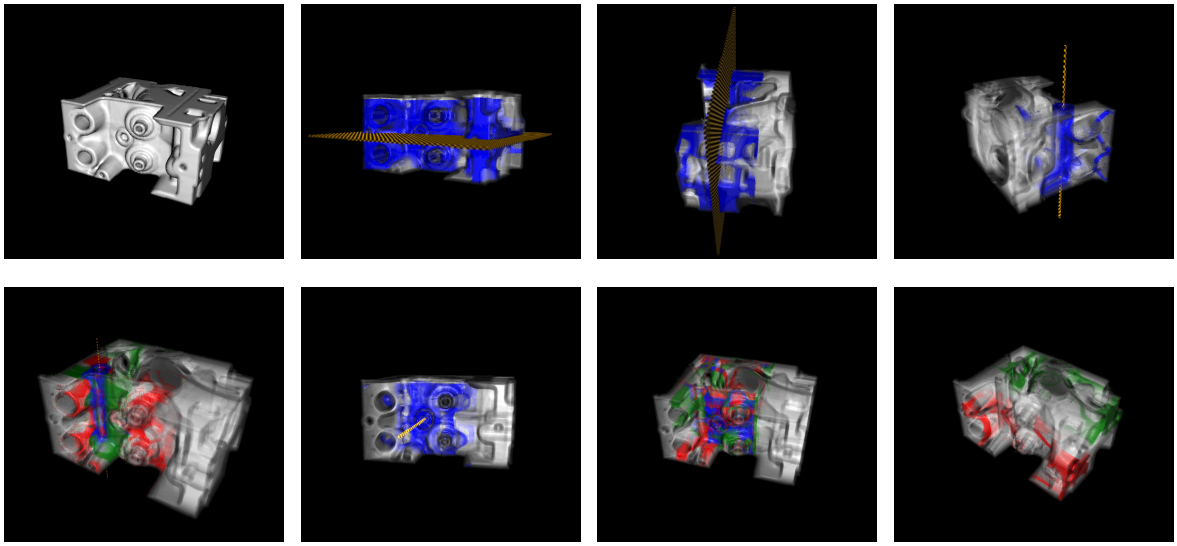


Figure 4: *Results of our method for the engine block data set (same color scheme as Fig. 3).*

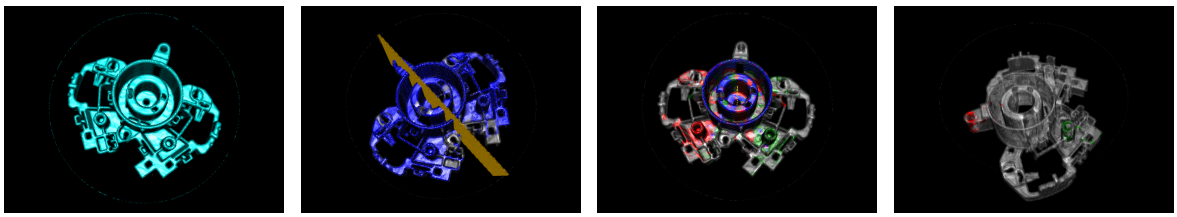


Figure 5: *A ct scan of a mechanical part and three extracted similarities (same color scheme as Fig. 3).*

Especially the last part of Figure 3 demonstrates that the detected symmetries are actually correct, but do not always correspond to the intuitively expected outcome. The reason for this is that matching subsets of the feature lines does not guarantee semantically meaningful correspondences in any case. In future work, this could possibly be addressed by penalizing orthogonal intersections of line feature graphs to be matched explicitly. This would reduce the ranking of such fragmented symmetries.

Since the sparse matching step only analyzes geometric constellations but the dense matching step compares actual intensity values, it is not necessarily the case that the highest ranked matching causes the largest set of corresponding voxels. Again, it would be possible to add an option to rerank a larger set of line-based matches by the actual volume covered.

4.2. Performance

The runtimes listed in Table 1 were measured on a dual Socket Intel Xeon X5650 2.66 GhZ with 48 GB of main memory. The basic unoptimized CPU implementation of our algorithm is written in C++ using OpenMP. We are convinced that an implementation on graphics hardware results in an interactive application.

The runtime for the feature detection step linearly scales with the resolution. In contrast to that, the sparse matching heavily depends on the complexity of the underlying geometry and the similarity therein. The VMV-Logo has a significantly smaller search universe but consumes the same time as the engine block model. In the synthetic case the iterative closest lines approach has to traverse very large segments of the model because many parts actually look the same. On the other hand there are a number of bases for the engine block which do not match at all and lead to a rejection very early. The timings for the dense matches are quoted for the generation of 20 symmetry volumes using the 20 best matches. They depend on the resolution and the amount of background voxels.

Model	$\lvert\mathcal{V}\rvert$	Detect	$\lvert\mathcal{B}\rvert$	Sparse	Dense
Cube	1 M	5 s	48	6 s	6s
Logo	1.9 M	11 s	684	206 s	11s
Engine	16.7 M	90 s	1602	208 s	74s
Part	7.1 M	42 s	2432	300 s	30s

Table 1: *Runtimes of the three algorithm stages for different models*

5. Conclusions and Future Work

We have presented an algorithm to find partial symmetries in 3D voxel data under Euclidean transformations. The algorithm transforms the volume to an intermediate representation consisting of line features that helps finding important symmetries efficiently and robustly. To the best of our knowledge, this is the first algorithm for detecting symmetries in volume data sets in these general settings.

In future work, we could address the problem of structuring the space of symmetries and employing this structure for visualization. In particular, we plan to augment our algorithm to detect hierarchical structures in the symmetries. Analyzing a video and detecting periodicities in time would be another interesting application. Developing a visualization technique that shows multiple different symmetries in a single volume rendering also constitutes a direction for future research. Unlike in the surface case, visualizing the symmetry structure of a dense volume data set in a single image is a very challenging task. In our experience, obvious solutions such as just coloring symmetric parts in the same color do not yield satisfactory results but frequently lead to visual clutter.

6. Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments on this manuscript. The engine block and toy car datasets are courtesy of *www.volvis.org* (in particular General Electric and University of Erlangen).

The work presented in this paper was made possible in part by the Cluster of Excellence “Multimodal Computing and Interaction” at the Saarland University, the the Intel Visual Computing Institute as well as the NIH/NCRR Center for Integrative Biomedical Computing, P41-RR12553-10 and by Award Number R01EB007688 from the National Institute of Biomedical Imaging and Engineering. The content is under sole responsibility of the authors.

References

[AMWW88] ALT H., MEHLHORN K., WAGENER H., WELZL E.: Congruence, similarity and symmetries of geometric objects. *Discrete Comput. Geom. 3*, 3 (1988), 237–256. 2

[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. 1, 2

[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). 1, 2

[BBW*09] BOKELOH M., BERNER A., WAND M., SEIDEL H.-P., SCHILLING A.: Symmetry detection using line features. *Computer Graphics Forum 28*, 2 (2009). 1, 2, 3, 4

[BWS10] BOKELOH M., WAND M., SEIDEL H.-P.: A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph. 29* (July 2010), 104:1–104:10. 2

[CJR07] CABAN J., JOSHI A., RHEINGANS P.: Texture-based feature tracking for effective time-varying data visualization. *IEEE Transactions on Visualization and Computer Graphics 13* (November 2007), 1472–1479. 2

[GCO06] GAL R., COHEN-OR D.: Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph. 25*, 1 (2006), 130–150. 1, 2

[GRVE07] GROTTEL S., REINA G., VRABEC J., ERTL T.: Visual verification and analysis of cluster detection for molecular dynamics. *IEEE Transactions on Visualization and Computer Graphics 13* (November 2007), 1624–1631. 2

[GSMCO09] GAL R., SORKINE O., MITRA N., COHEN-OR D.: iwires: An analyze-and-edit approach to shape manipulation. *ACM Trans. Graph. 28*, 3 (2009). 2

[HLRS*08] HADWIGER M., LAURA F., REZK-SALAMA C., HOLLT T., GEIER G., PABEL T.: Interactive volume exploration for feature detection and quantification in industrial ct data. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (nov.-dec. 2008), 1507 –1514. 2

[HS08] HONG Y., SHEN H.-W.: Parallel reflective symmetry transformation for volume data. *Computers & Graphics 32*, 1 (2008), 41–45. 2

[KBW*10] KERBER J., BOKELOH M., WAND M., KRÜGER J., SEIDEL H.-P.: Feature preserving sketching of volume data. In *Proceedings of the 15th International Workshop on Vision, Modeling and Visualization* (November 2010), Koch R., Kolb A., Rezk-Salama C., (Eds.), Eurographics Association, pp. 195–202. 3

[LBM*06] LANEY D., BREMER P. T., MASCARENHAS A., MILLER P., PASCUCCI V.: Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics 12* (September 2006), 1053–1060. 2

[LE06] LOY G., EKLUNDH J.: Detecting symmetry and symmetric constellations of features. In *Proc. Europ. Conf. Computer Vision* (2006), pp. 508–521. 1, 2

[Low03] LOWE D.: Distinctive image features from scale-invariant keypoints. In *Int. J. Computer Vision* (2003), vol. 20, pp. 91–110. 2

[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. 1, 2

[MSHS06] MARTINET A., SOLER C., HOLZSCHUCH N., SILLION F.: Accurate detection of symmetries in 3d shapes. *ACM Trans. Graph. 25*, 2 (2006), 439 – 464. 2

[OSG08] OVSJANIKOV M., SUN J., GUIBAS L.: Global intrinsic symmetries of shapes. In *Eurographics Symposium on Geometry Processing (SGP)* (2008). 1, 2

[PMW*08] PAULY M., MITRA N. J., WALLNER J., POTTMANN H., GUIBAS L.: Discovering structural regularity in 3D geometry. *ACM Trans. Graph. 27*, 3 (2008). 1, 2

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A planar-reflective symmetry transform for 3D shapes. *ACM Trans. Graph. 25*, 3 (2006). 1, 2

[SBM*10] STAVA O., BENES B., MECH R., ALIAGA D. G., KRISTOF P.: Inverse procedural modeling by automatic generation of l-systems. *Comput. Graph. Forum 29*, 2 (2010), 665–674. 2

[SKS06] SIMARI P., KALOGERAKIS E., SINGH K.: Folding meshes: hierarchical mesh segmentation based on planar symmetry. In *Proc. Symp. Geometry Processing* (2006), pp. 111–119. 1, 2

[SMB10] SCHULZE F., MAJOR D., BÜHLER K.: Fast and memory efficient feature detection using multiresolution probabilistic boosting trees. In *roceedings of the 19th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)* (2010), vol. 19 of *Journal of WSCG*, pp. 33–40. 2

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Proceedings of the Sixth International Conference on Computer Vision* (Washington, DC, USA, 1998), ICCV '98, IEEE Computer Society, pp. 839–846. 3

[TM05] TZENG F.-Y., MA K.-L.: Intelligent feature extraction and tracking for visualizing large-scale 4d flow simulations. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing* (Washington, DC, USA, 2005), SC '05, IEEE Computer Society, pp. 6–. 2

[TW05] THRUN S., WEGBREIT B.: Shape from symmetry. In *Proc. Int. Conf. Computer Vision* (2005). 2

[VNP*05] VIJAY A. G., NATARAJAN V., PASCUCCI V., TIMO BREMER P., HAMANN B.: Topology-based simplification for feature extraction from 3d scalar fields. In *In Proc. IEEE Conf. Visualization* (2005), pp. 535–542. 2

[WCHK03] WU Y., CHANG E. C., HUANG Z., KANKANHALLI M. S.: Feature extraction of volume data based on multi-scale representation. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (New York, NY, USA, 2003), GRAPHITE '03, ACM, pp. 175–180. 2