Symmetry Detection in Large Scale City Scans

Jens Kerber, Martin Bokeloh,
Michael Wand, Hans-Peter Seidel

MPI-I-2012-4-001 April 2012

Authors' Addresses

Jens Kerber
Max-Planck-Institut für Informatik
Campus E 1 4
D-66123 Saarbrücken

Michael Wand
Max-Planck-Institut für Informatik
Campus E 1 4
D-66123 Saarbrücken

Martin Bokeloh
Stanford University
353 Serra Mall
Stanford, CA 94305, USA

Hans-Peter Seidel
Max-Planck-Institut für Informatik
Campus E 1 4
D-66123 Saarbrücken

Abstract

In this report we present a novel method for detecting partial symmetries in very large point clouds of 3D city scans. Unlike previous work, which was limited to data sets of a few hundred megabytes maximum, our method scales to very large scenes. We map the detection problem to a nearest-neighbor search in a low-dimensional feature space, followed by a cascade of tests for geometric clustering of potential matches. Our algorithm robustly handles noisy real-world scanner data, obtaining a recognition performance comparable to state-of-the-art methods. In practice, it scales linearly with the scene size and achieves a high absolute throughput, processing half a terabyte of raw scanner data over night on a dual socket commodity PC.

Keywords

symmetry detection, feature detection, large scene processing, clustering

Contents

1 Introduction **2**

2 Symmetry Detection **5**
2.1 Scalable Symmetry Detection 6

3 Implementation **8**
3.1 Data Organization 8
3.2 Line Features and Sample Points 9
3.3 Descriptor 11
3.3.1 Line Feature Images 12
3.3.2 Orientation Histogram 12
3.4 Clustering 13
3.4.1 Rapid Geometric Alignment 13
3.4.2 Geometric Clustering 14
3.5 Dynamic Area Queries 15

4 Implementation and Results **16**
4.1 Parameters and Performance 16
4.2 Full Hannover Data Set 23
4.3 Discussion 25

5 Conclusions and Future Work **26**

1 Introduction

Symmetry detection [17, 36, 47, 50, 46, 41, 8, 26, 38] has become an important tool for the analysis of digital 3D shape models. A symmetry detection algorithm examines an input 3D model for global or partial self-similarities. This means that portions of an object match back to itself under a constraint class of mappings, such as rigid motions or similarity transforms. The output is a list of parts and corresponding transformations that map these parts non-trivially back onto the original surface.

In other words, symmetry detection discovers structural regularity and redundancy in shapes. This information is essential for a large and growing number of applications: The discovered redundancy can for example be utilized for compression [36] and data cleanup by averaging matching parts [18, 46, 8, 56]. Furthermore, symmetry-based invariants have been exploited for symmetry preserving editing [19, 9, 54, 57], for semi-automatic creation of related shapes [37, 10, 25], and to reason about the functionality of shapes [39].

Symmetry detection algorithms published so far all share an important limitation: Current approaches do not scale to large quantities of geometry. The largest scenes for which results have been reported in literature are in the range of a few hundred megabytes in size [36, 46, 8]. However, there are application scenarios where data has to be handle that is several orders of magnitude larger. For example, there are ongoing large scale urban scanning campaigns (such as Google street view, which includes 3D laser scanning), producing enormous quantities of 3D point clouds depicting most urban areas. For such massive geometric data sets, discovering redundancy through symmetry analysis is even more relevant than for small scenes. In addition to the obvious need for identifying redundancy and obtaining a compact encoding, applications such as non-local scan consolidation or structure learning approaches would clearly benefit from a larger data base and could be expected to perform better. However, available symmetry detection algorithms are fundamentally limited in the input size that can be handled.

Many previous techniques are based on transformation voting [36]. These approaches consider pairs of points with matching local neighborhoods, compute a canonical transformation between them, and vote for them in transformation space, disregarding the spatial location. For a large scene with many symmetric elements, the transformation space is cluttered by the overlay of simultaneous matches and the structured noise that comes with them. This is why the discrimination of various symmetries becomes increasingly more difficult. Analyzing a city-scale scan in a single transformation space is clearly infeasible; even larger buildings already require a hierarchical decomposition for disambiguation [36].

An alternative are approaches that match local feature constellations [6, 8]. These techniques are more restricted as they require the presence of suitable features in the model. However, they can handle large amounts of partial symmetries with arbitrary structure. Nevertheless, the feature-based approaches essentially work by comparing all pairs of feature constellations explicitly. Some speedup is obtained by randomized sampling but the scaling behavior is quadratic in the size of the scene. Furthermore, all of the processing is performed in-core and with random access to the data so that the potential scene size is limited by available main memory. At moderate scene sizes, this is not yet an issue, but handling very large data sets is not possible. Other techniques, such as computing robust pairwise auto-alignments [50] or moment analysis [32] are also not applicable to large scenes.

In this paper, we make a first step towards symmetry detection in very large scenes. We present a new, scalable symmetry detection algorithm that is designed to handle large amounts of input geometry. In order to obtain a scalable algorithm, we need to avoid explicit pairwise comparisons of matching geometry, which leads to (unacceptable) quadratic complexity. Rather than that, we find descriptors of local geometry that are directly mapped into a feature space such that similar geometry coincides. By aggregating information in an appearance rather than a transformation space, it is easier to discriminate a large number of symmetric parts.

Algorithmically, our approach is in line with several recent algorithms that aim at finding similarity in very large quantities of data through clustering (such as clustering of images [27, 15]). The mapping can be done in a single, linear scan, allowing for external data storage on hard drive. Afterwards, a clustering step is performed to match nearby descriptors. The efficiency of this step strongly depends on the design of the descriptor space. The trade-off is to find a low-dimensional representation that retains as much characteristic information of the shape as possible and is robust to noise and missing data.

The reason for seeking a low-dimensional representation is the "curse of dimensionality": Clustering requires retrieving similar descriptors, i.e., proximity queries in descriptor space. For high dimensions of this space, most traditional exact and even approximate range query techniques become inefficient. There are approaches like locality sensitive hashing that can achieve sub-linear query times in high dimensions [3], relying on the Johnsen-Lindenstrauss Lemma [14]. However, these techniques come at considerable costs in terms of runtime constants and super-linear memory requirements that again impede scalability [22]. In summary, the more we are able to discriminate geometry by descriptors with few degrees of freedom, the easier and more efficient the search problem (and thereby the clustering) becomes.

We propose a descriptor of local geometry that has been designed with these goals in mind: We combine the idea of gradient histograms [29, 13] with local crease-line detection [40, 8], both of which have been proven to be very effective in characterizing local geometry. We cluster the resulting descriptors using an additional geometric verification step. This step performs a linear number of pairwise comparisons to disambiguate different pieces of geometry with similar descriptors. In order to keep absolute processing times small as well, we introduce for this task a novel, very efficient geometry matching algorithm, based on ideas of real-time viewfinder alignment [1]. As a result, we obtain an algorithm that can detect partial symmetries in very large data sets efficiently both in terms of asymptotic scaling behavior as well as in terms of absolute performance.

We evaluate the accuracy and performance of our proposal on real-world 3D scans, suffering from noise, outliers, and incomplete acquisition. For example, for the well known "Hannover" city scans, by Brenner et al. [11], with 12GB of raw data size, we can find the partial symmetries within the whole scene in less than 70 minutes on dual-socket commodity PC. Nevertheless, we maintain a recognition accuracy comparable to the previous technique by Bokeloh et al. [8], which is probably the most scalable technique for this task but still is not able to compute a result on the full data set. In order to study scalability further, we create a larger scene by replicating parts of the Hannover scan, creating up to 500GB of raw data. Even in this case, symmetry detection can be performed over night on a single PC.

2 Symmetry Detection

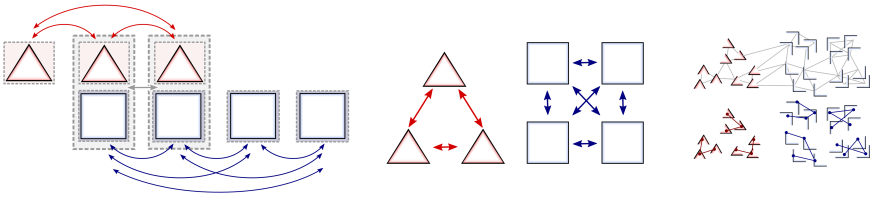
Before discussing our approach in more detail, we would first like to formalize the notion of partial symmetry detection so that it becomes clear what we are aiming at computing and why it is possible to speed this computation up.

A symmetry detection algorithm takes a piece of geometry $\mathcal{S} \subset \mathbb{R}^3$ as input and considers a group of transformations $\mathcal{T}$ acting on it, which is a subgroup of the bijections of $\mathbb{R}^3$. Most frequently, these are rigid motions $\mathcal{T} = E(3)$ [8], or similarity transforms [36]. In our paper, we consider a restricted class of true rigid motions $SE(3)$ (excluding reflections) that keep a global upward direction $\mathbf{u}$ of the scene invariant, i.e., $\mathbf{Tu} = \mathbf{u}$ for all $\mathbf{T} \in \mathcal{T}$. This is only a minor restriction for architectural models but simplifies the construction of our descriptors substantially.

We now consider all subsets of $\mathcal{S}$ that are mapped back to $\mathcal{S}$ under transformations $\mathbf{T} \in \mathcal{T}$: The set of *pairwise correspondences* $\mathcal{C}$ with respect to input $\mathcal{S}$ and a transformation group $\mathcal{T}$ is defined as:

$$\mathcal{C}(\mathcal{S}, \mathcal{T}) = \{(\mathcal{P}, \mathbf{T}) | \mathcal{P} \subseteq \mathcal{S}, \mathbf{T} \in \mathcal{T}, \mathbf{T}(\mathcal{P}) \subseteq \mathcal{S}\}$$

A single correspondence consists of a subset $\mathcal{P}$ of the scene $\mathcal{S}$ and a transformation $\mathbf{T} \in \mathcal{T}$ that matches this piece to an alternative location. Obviously, there are a large number of such correspondences: For example, each subset of $\mathcal{P}$ itself is again a valid correspondence under $\mathbf{T}$. We can remove the ambiguity by considering only maximal sets $\mathcal{P}$. Another issue are continuous symmetries: For example, a plane has an infinite set of symmetry transformations that map portions of it back to itself. To avoid computational costs, such continuous symmetries are usually computed a priori using a differential analysis [20] and can be excluded from the further detection [10]. Our algorithm follows this idea and excludes continuous symmetries by only computing matches at non-slippable features [8]. Finally, we also do not want to enumerate spurious matches that lead to very small subsets $\mathcal{P}$; a minimal requirement should be that the detected parts $\mathcal{P}$ have a certain minimum



(a) input (b) pairwise cliques (c) descr. graph

Figure 2.1: Utilizing the structure of partial symmetries: (a) An input scene containing partial symmetries. (b) Symmetric subsets form cliques with a quadratic number of partial correspondences. In the exact case, only a linear number of comparisons is required to retrieve them. (c) We use approximate descriptor matches and walk a neighborhood graph to find the symmetry cliques of feature points.

extend. Our method implements this filter by demanding a match of local neighborhoods for matching points.

Making all of the mentioned restrictions still does not solve the complexity problem of symmetry detection: We still obtain a large number of overlapping subsets $\mathcal{P}$ that map back to $\mathcal{S}$ under various transformations $\mathbf{T}$. Explicitly computing all of these sets by a pairwise matching of geometry is possible for small scenes [10] but prohibitively expensive for large inputs.

However, a computation of all pairwise matches is not required, as illustrated in Figure 2.1: Quadratic complexity is caused by overlapping matches. If we consider a single point on the model, the set of all points reached by a symmetry transformation forms a clique [26]. This is because correspondence, as we have defined it, is an equivalence relation. Consequently, it is sufficient to only walk a linear-sized subgraph of the full clique to discover all symmetric points.

2.1 Scalable Symmetry Detection

In our paper, we implement this idea in an approximate setting: First, we restrict the scene to feature points $\mathcal{F} \subseteq \mathcal{S}$ that are non-slippable, i.e., not part of a continuous symmetry. Then, for each feature point $\mathbf{x} \in \mathcal{F}$, we compute a descriptor vector $descr(\mathbf{x}) \in \mathbb{R}^d$. The descriptor maps the local r-neighborhood $N_r(\mathbf{x}) := \{\mathbf{y} \in \mathcal{S} | dist(\mathbf{x}, \mathbf{y}) < r\}$ of points $\mathbf{x} \in \mathcal{S}$ to a short vector that summarizes the local geometry. This mapping is of course not injective, but we will construct a descriptor that ensures that similar geometry maps to similar descriptor values while dissimilar geometry with high likelihood obtains dissimilar descriptors.

In this descriptor space, we build a k-nearest-neighbor graph (typically $k = 10..50$). By walking along this graph (Figure 2.1c), we will with high likelihood encounter matching geometry. Using k nearest neighbors ensures that both approximate matches and overlaps in descriptor space are handled correctly: As long as k is large enough, we will be able to retrieve the whole fully connected clique by only performing a linear number of comparisons. We will show empirically that our choice of descriptor is discriminative enough to ensure high recall rates at small values of k. The actual pairwise comparisons are performed on geometry rather than descriptors: We use a two-step cascaded test to compare surface crease lines; the first step of the cascade is almost as accurate as matching the full point sets but up to four orders of magnitude faster.

Our algorithm outputs for each feature point its clique of matching points. This information is only linear in size and encodes all correspondences $\mathcal{C}(\mathcal{S}, \mathcal{T})$ of the whole scene implicitly. Unlike traditional approaches (such as [36, 8]), we do not segment the scene into symmetric pieces, but output the matching for each point [10]. This representation provides more information than an ad-hoc partitioning [38] as it encodes all overlapping symmetries.

In the subsequent section, we discuss in detail how we implement this concept in practice.

3 Implementation

We now discuss the processing pipeline of our symmetry detection algorithm. Figure 3.1 depicts the interplay of the different stages. Before anything else, we need to prepare and organize the raw data, as detailed in Section 3.1. After that, the first processing stage of the pipeline extracts a crease-line abstraction of the geometry and sample points for the later matching step. This process is explained in Section 3.2. In order to match these keypoints by direct mapping, we compute a suitable descriptor for each sample point, as summarized in Section 3.3. The detection of symmetries is then reduced to a clustering problem that uses descriptor information in conjunction with a rapid, approximate geometry matching algorithm for verification (Section 3.4).

3.1 Data Organization

The input to our algorithm is a raw point cloud from a 3D scanner, stored as a long list of 3D points, potentially amounting to several hundreds of gigabytes of data. Our first task is to organize this data for efficient further processing. We use the data structure of Wand et al. [53], who provide an open source implementation on the web. For completeness, we briefly discuss the main steps.

The first task is to partition the data into blocks that fit into main memory. This is done by a hashing scheme: The scene is overlayed with a regular grid and each grid cell is associated to a file on the hard disc. Then, in a single scan through the input data, the point set is split up into manageable parts. After that, an octree is build to organize the pieces hierarchically, with the leave nodes storing the subsets of the input data. Each leave contains at most 64K points, which is the block size for the further processing.

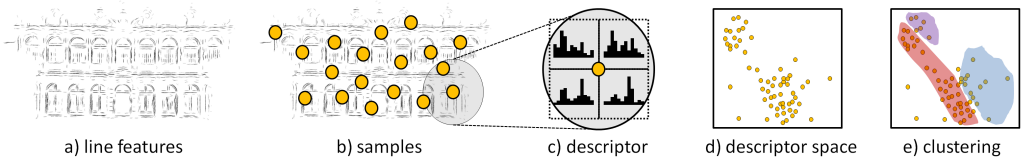


Figure 3.1: Overview: First, we extract line features (a) and a set of sparse samples (b) from the input. Then, we compute a high-dimensional descriptor (orientation histograms) for each sample (c) and project all descriptors to a low-dimensional space (d). Now, we use the k-nearest-neighbor topology in descriptor space and cluster samples by region growing (e), which is performed by a pairwise alignment and comparison of the line feature representation. Distances in descriptor space serve only as a filter to reduce the complexity, not as a final matching criterion.

The employed system in addition also builds a multi-resolution rendering data structure that is not necessary for the processing but which we use for visualizing the results. We also use the provided processing system to estimate normals by local PCA fitting and orient them according to the (assumed to be) known scanner positions, giving us an unambiguous surface orientation everywhere.

3.2 Line Features and Sample Points

The first goal of our processing pipeline is to reduce the amount of data to a more manageable set while maintaining most of the important shape information. We follow the previous work of Bokeloh et al. [8] and compute a line-feature representation that extracts the salient crease lines from the input geometry as an abstraction of the full point cloud. Crease lines can capture the most important part of the shape structure at small representational costs: From only sparse crease lines, a surprisingly exact surface representation can be reconstructed [40].

In practice, slippage-based line features [8] yield a storage reduction of about a factor of 170 (from 12GB down to 72MB for the “Hannover” data set). Assuming that at least 8GB of main memory are available in most of-the-shelf PCs today, this implies that this representation can compress the original input data of more than a Terabyte to still fit into main memory, where more flexible processing is possible.

We implement a parallel out-of-core algorithm for computing the line features efficiently: The feature extraction method in [8] retrieves points with linear slippability [20], corresponding to a general notion of line features. Afterwards, a projection operator is used that draws surface points onto the closest point of maximal curvature in direction orthogonal to the line direction. To compute the set of line features from the input, sample points are randomly distributed over the surface using Poisson disk sampling with spacing σ and each sample is projected individually onto a line segment, never moving by more than σ. Since every projection is independent from other samples we can process many sample points in parallel.

Parallel out-of-core implementation: We implement this strategy in a data-parallel, out-of-core fashion: From the octree, we extract all child nodes, which we subsequently treat as processing *blocks*. For each child node, we retrieve the direct neighbors to provide context at the boundaries and cut out an extended region, as follows: The projection operator requires a small local neighborhood of surface points (a sphere with radius σ) in order to compute surface properties such as curvature or slippage analysis [21]. Furthermore, a projection can move a point up to σ away from its original position. This means for each process block we need to add all points within a 2σ boundary to the processing block in order to compute correct projections. Further, in order to maintain a uniform sampling density of line features that is not affected by process block boundaries, we remove all line features outside their initial process block boundary.

We now use a single thread to load these blocks from disc and extract the neighborhoods. We traverse the tree coherently (depth first) and use an LRU cache for blocks. The extraction thread then dispatches work packages to several processing threads that perform the line feature extraction for several blocks in parallel. To speed-up the range queries of the projection algorithm, local spatial hierarchies are used. As the computational costs for the projection are much higher than for just loading and clipping the data, we usually obtain full CPU utilization on a our reference machine (12 cores with 24 threads); only for small scenes, we sometimes observe a suboptimal CPU utilization because the load-balancing is quantized to whole blocks.

Feature points: In addition to line features, we extract a relatively sparse set of sample points from the input that we will equip with descriptors later on, and for which we will actually compute symmetry information. We call these points *feature points* in the following. We use junctions of crossing line features (called "bases" in [8]) as samples and remove points that are too close to each other, again by Poisson disc sampling.

Our descriptors will cover a radius r, which is typically 20σ. The descriptors are designed with robustness towards local shifts so that we can rather aggressively thin out the sample set; we currently use a sample spacing of 20% of the descriptor radius r .

3.3 Descriptor

From the construction of the feature point, we can assume that matching pairs of symmetric instances will contain sample points that are nearby with respect to the symmetry transformation but vary in their exact position. Therefore, we have to ensure that the descriptor is invariant under translations up to a tolerance of at least the sample spacing ($0.2r$).

Such invariance is a common problem in computer vision, where robustness towards small transformations as well as local distortions is crucial for effective image descriptors. A widely employed and well-performing standard solution to this problem is the use of gradient orientation histograms [29, 13, 33]. If we were comparing images by simple template matching, the robustness against local distortions and shifts can be improved by low-pass filtering. However, comparing blurred images destroys a large amount of information and therefore significantly impacts the precision of the results. In order to have good recall under distortions without compromising discrimination, histograms of oriented gradients (HoG) are used. Instead of averaging color, a histogram of the orientation of local image gradients is build, which itself already gives a remarkably discriminative descriptor [16]. In other words, spatial averaging is necessary for robustness, and HoG-type descriptors perform this averaging by building orientation histograms rather than averaging color, which retains much more information while still providing the same local shift invariance.

This idea can be transferred to the geometric domain by using crease lines on the geometry as the analogon of image gradients, both of which contain the most important information about the data. This idea has been examined for mesh data as "mesh SIFT" [31], leading to favorable results. Our situation is different as we are dealing with noisy and partial point cloud data; and we targeting large data sets and thus aiming at high throughput. We build our algorithm upon the previously computed sparsely sampled line feature representation. We examine the distribution and constellations of line feature directions in local neighborhoods to build orientation histograms. Figure 3.2 illustrates the concept.

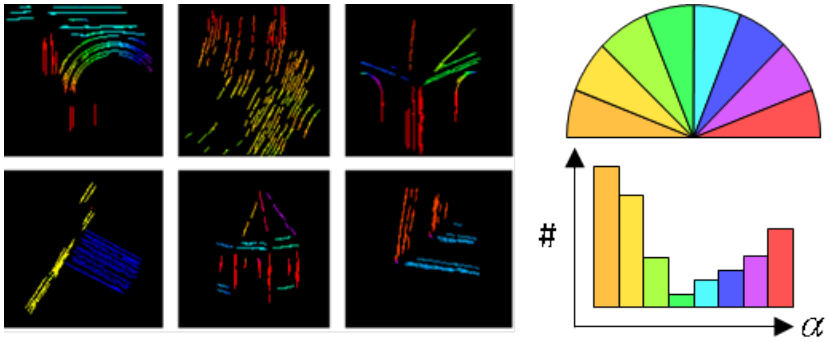


Figure 3.2: Examples of 6 different line feature images from the Hannover data set. Directions are indicated by different colors. Orientation histograms capture the distribution of line directions without considering spatial location.

3.3.1 Line Feature Images

As motivated in Section 2, we assume that we have an urban scene with a consistent upward orientation $\mathbf{u}$. Mappings between symmetric instances cannot change this upward direction. To build descriptors, we consider all line features in a sphere of radius r around each sample point. For this set of features, we fix a local coordinate frame aligned with the upward orientation and the average surface normal that we approximate by performing principal component analysis on the line features. From our experience, it is sufficient to consider the projection of line features onto the plane orthogonal to the average surface normal because architectural facades mostly resemble reliefs with relatively small depth. As the projection of matching feature points is consistent, our descriptor will still work for objects with arbitrary depth, but the overlay of features might be a bit less discriminative. The 2D projections of the line features form *line feature images*, from which we derive the descriptors. We use 128×128 pixel images, in which we render the computed line samples by drawing short segments of length σ.

3.3.2 Orientation Histogram

The orientation histograms encode the distribution of orientations discretized into a set of orientational bins. In this work, we use a relatively low resolution of typically $B = 8$ different (undirected) angles which has the advantage of compactness and also acts as a low pass filter of orientations under the assumption of random noise in the line feature orientations.

In order to increase the expressiveness of our descriptors, we compute multiple orientation histograms with overlapping but different spatial regions, on a $H \times H$ grid. In practice, we have used 4×4 histograms to achieve all the results shown within this paper. Our evaluation in the upcoming section will justify this parameter setting. To avoid aliasing, we use a Gaussian Window function with standard deviation r/H in the spatial domain and $180°/B$ degree in the orientation domain and distribute contributions to overlapping bins correspondingly. As shown by Dalal et al. [13], such a proper filtering is instrumental to good results.

The number of dimensions increases with spatial and orientational binning. We therefore use principal component analysis to project to a lower dimensional basis with D dimensions. We will later show experimentally that $D = 8$ is a good choice where the dimensionality reduction has only minimal impact on recognition performance.

3.4 Clustering

We now use the computed descriptors to cluster matching geometry, thereby computing the symmetry transformation of each feature point. Two points $\mathbf{x}, \mathbf{y} \in \mathcal{F}$ are considered symmetric, if there exist a $\mathbf{T} \in \mathcal{T}$ such that $\mathbf{T}(N_r(\mathbf{x})) = N_r(\mathbf{y})$. By construction of the descriptors, we know that this implies $descr(\mathbf{x}) = descr(\mathbf{y})$, but not the other way round. Similar geometry will have similar descriptors, but multiple different pieces of geometry can still map to the same descriptor. Therefore, we need an additional verification step for effective clustering. The key idea is to consider nearby descriptors as matching candidates and verify the quality of the match by an explicit geometric alignment.

3.4.1 Rapid Geometric Alignment

At this point, we need a fast algorithm to align two feature line images. Obviously, we could use the iterative clostest lines (ICL) algorithm originally employed in [8]; however, the absolute costs are high. We therefore propose an efficient approximation that is at least two orders of magnitude faster (in a practical experiment, the run time of rapid alignment decreased to 4 minutes from 480 minutes for standard ICL). The new technique also has the advantage of a larger convergence radius, as it performs a global search (versus the local search of ICL).

We adapt the technique from Adams and co-workers [1] for rapid image alignment: We project all line features with a horizontal orientation on the vertical axis and compute the distribution of line features along that axis. We call the result an *alignment histogram*. In order to align two images in the vertical direction, we compute the correlation between both alignment histograms and take the maximum correlation as alignment. Horizontal alignment is achieved accordingly. Rotational alignment is given a priori by the coordinate frame defined by upward direction and mean normal (which is robust as it has been averaged over a large area). In contrast to the original approach, we perform the correlation in the Fourier domain using FFT, for further speed-up.

With the estimated alignment we now compare the line feature images by comparing shifted images, which we compute on the fly to save memory. Comparison is done by normalized cross-corellation (NCC), and after blurring with a small Gaussian kernel that again avoids aliasing and sensitivity to alignment errors that are expected to be in the range of one or two pixels. Please not that the images are gray-scale for NCC starting from binary lines before blurring.

3.4.2 Geometric Clustering

Our clustering algorithm is a simple region growing technique: We consider a graph with feature points as nodes and edges connecting the k-nearest-neighbors in descriptor space (using Euclidean distance). The edges connect different candidates of matching geometry, which we now verify during the clustering: We traverse the graph breadth first. For each new node encountered, we compute an alignment of the feature line images of the new node to that of the original start node using the rapid alignment technique. If and only if the images match (i.e., the NCC score is above a user chosen threshold ϵ), we continue the search towards the unprocessed neighbors of this node. Please note that we always compare to the start node to avoid drift. It is also important to stress that we utilize two comparison measures at this point: On the one hand, similarity of descriptors serves as initial filter to obtain candidate matches. On the other hand, the actual match of geometry is used as criterion to steer and stop the traversal of the descriptor graph.

Filtering by descriptors reduces the number of pairwise comparisons from naive $O(n^2)$ to $O(kn)$ for n samples. If the descriptor is effective, k is a small number. We will show empirically that small values of $k = 10..50$ are sufficient to obtain good results on complex scenes with a large amount of geometric variation.

The only performance critical part that remains is the computation of nearest-neighbors in descriptor space. The design of our descriptors allows us to use rather low-dimensional feature vectors with only 8 dimensions. In addition, the reduction to sparse samples permits storing samples in-core even for very large scenes. We therefore currently employ the approximate nearest-neighbor library (ANN) by Mount and Arya [5], which performs well for spatial queries with medium dimensionality.

3.5 Dynamic Area Queries

The clustering algorithm described above concludes the preprocessing of symmetry information. The output is a partition of the set of all feature points into fully connected clusters of matching geometry.

We now discuss how this information can be utilized to derive more precise symmetry information. We assume that the user (as in our experiments) or an outer-loop algorithm (in potential applications) wants to retrieve all pieces of geometry within the whole scene $\mathcal{S}$ that is symmetric to a given query piece $\mathcal{P} \subset \mathcal{S}$. In this situation, we have stronger cues because we can combine the information of *all* feature points $\mathcal{F}_\mathcal{P}$ within $\mathcal{P}$ *simultaneously*.

We first enumerate all clusters the feature points in $\mathcal{F}_\mathcal{P}$ belong to. From this, we retrieve a set of transformations to symmetric instances. Since the clusters can still contain outliers (depending on the NCC threshold), we align all line features within the query region to the potential symmetries using iterative closest line (ICL) [8]. We use the resulting ICL score to validate a symmetry and display the symmetry if at least 50 percent of all line features find a correspondence.

We recursively repeat this process with the transformations we can find in the symmetric regions we match to, i.e., building the transitive closer via region growing and verification by ICL to the original instance $\mathcal{P}$. As shown below, this transitive-closure step significantly improves the results.

In our experiments, we always let the user chose rectangular bounding boxes to cut out example regions $\mathcal{P}$; in the corresponding images these are shown as wire-frame boxes. Please note that the region queries are dynamic, not precomputed (therefore, the costs are not included in the processing costs given). The queries always search the whole scene, not only the visible portion. Nevertheless, response times are interactive (in the range of a few second for the full "Hannover" test scene).

4 Implementation and Results

We now evaluate our method in two steps: First, we study the influence of algorithmic and complexity parameters. Second, we apply our method to real-world scenes and demonstrate some practical results. All experiments were performed on a dual Socket Intel Xeon X5650 2.66 GHZ (6 physical cores + hyperthreading) equipped with 48 GB of main memory and an NVIDIA GeForce GTX 480. The code for line feature extraction uses multi-threading with 24 threads, the remainder of the code is currently single threaded, as feature extraction dominated the runtime.

Benchmark data: As benchmark data set, we have chosen the Hannover city scan collection, provided by Brenner et al. [11]. This data set features a diverse collection of many different buildings all over the city of Hannover, Germany, collected by 3D time-of-flight scanning and subsequent scan registration. As a real-world data set, it contains significant noise artifacts, as well as clutter and outliers. Furthermore, many of the buildings are acquired only partially and the sampling density varies considerably (as it can be seen in the accompanying video). The full collection consists of 463 million points, or roughly 14GB in binary format. As we are not aware of publicly available data sets of larger size, we use replications of parts or all of the data to study the scaling behavior for even larger scans.

4.1 Parameters and Performance

Our method has a number of parameters that might significantly affect the practical performance. Therefore, we first systematically study the influence of these parameters. We proceed in two steps: First, we only look at the recognition performance of the descriptor and chose good parameter values. Second, we study the full pipeline and its behavior under different configurations.

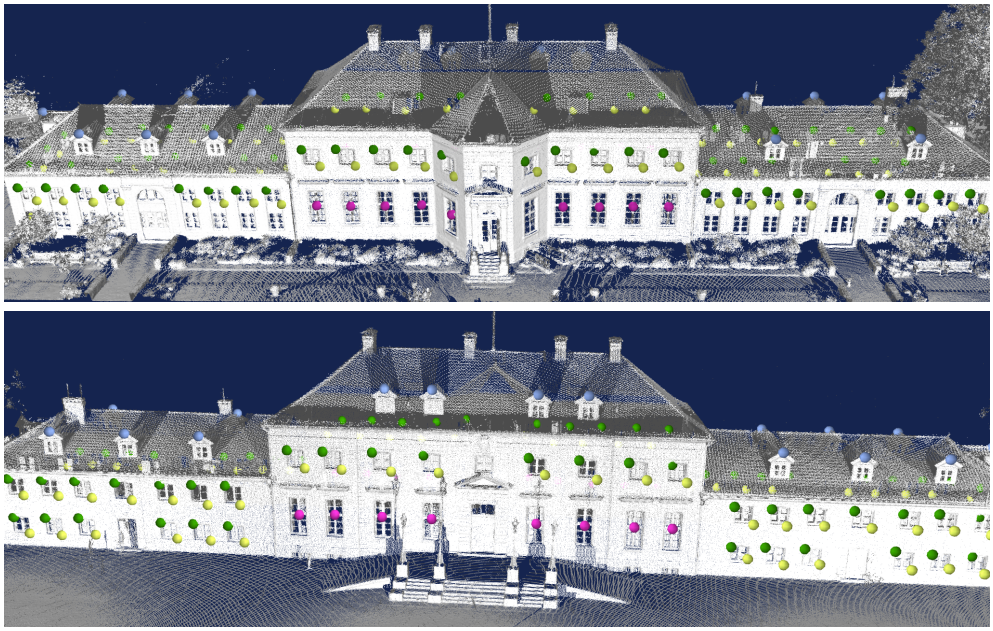


Figure 4.1: Annotated benchmark data. All spheres with the same color indicate symmetric geometry.

Benchmark setup: We have chosen the "Wilhelm Busch museum", which is part of the Hannover scans, and manually annotated the data: We label points on the objects surface with symmetry classes (see Figure 4.1). These annotations capture the four most important symmetry classes; our test ignores other matches.

Descriptor test: We now first test the descriptor performance. For this test, we retrieve the sample point closest to each annotated point and match it against all other samples using their descriptors. A good descriptor should match these descriptors to other descriptors close to an annotation point of the same class, and only to them. We define closeness by a sphere of radius $0.5r$, which is the maximum distance at which our alignment scheme could possibly find a match of the descriptor images. According to this definition, we compute false negative rates (averaged over all features and descriptors), and the absolute number of false positives, normalized by dividing by the number of tests (i.e., overall number of annotated points).

The results are shown in Figure 4.2. The curves correspond to different threshold values in matching the descriptors. We first test different parameters for the number of spatial histograms H (Figure 4.2a).

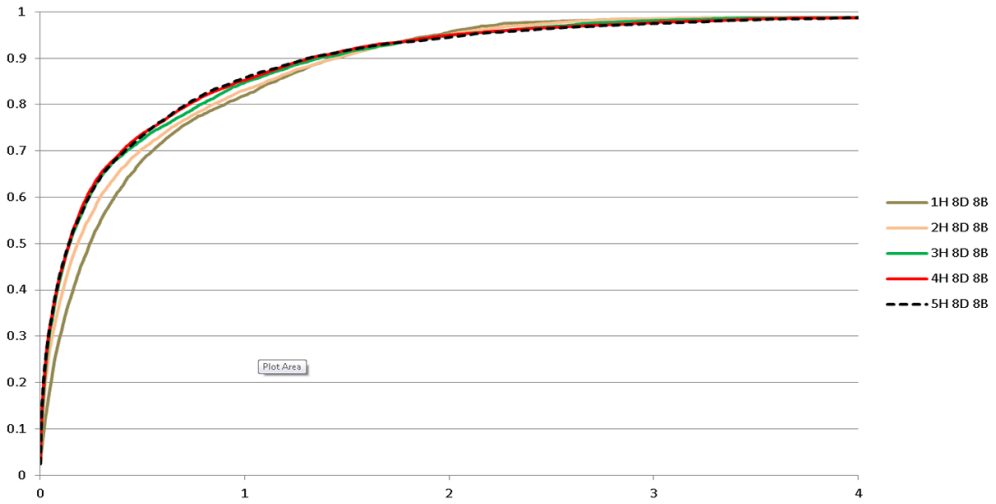
It turns out that 4×4 spatial histograms yield a good performance, with only marginal improvements for larger values.

Next, we examine the effect of dimensionality reduction (Figure 4.2b) and the influence of the number of orientation histogram bins. We keep the number of histograms constant at $H = 4$. For orientation histograms, dividing the angular range of 180° into 8 bins leads to the best results. 16 bins as well as 4 bins are worse. The reduction from 16 to 8 dimensions via PCA impedes the recognition rate only marginally, while providing substantial benefits for an efficient nearest-neighbor search. In summary, these test justify our parameter choice with which we achieved all the results shown below (4×4 histograms with 8 orientation bins overall reduced to 8 dimensions). We want to stress that the descriptors for all our experiments are normalized. Normalization means that we scale the histogram entries by the inverse of the l_2 norm of the full histogram before PCA reduction. This significantly improves the recognition performance.

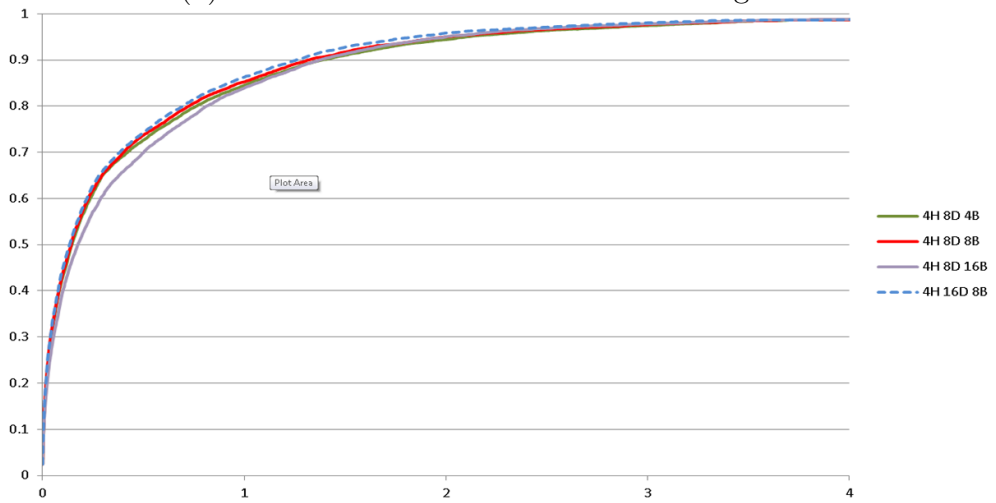
Full pipeline benchmark: We now test the full pipeline, including the rapid geometric alignment. We use again the annotated data for the Museum scene and apply the same criteria as in the previous test, with a tolerance radius for a match of $2r$. For counting true and false positives, we use two separate criteria: First, we just evaluate the accuracy of the pairwise matches, as done before. In addition to that, we also examine the transitive closure of the matches, which is also done by dynamic query algorithm (Section 3.5).

Figure 4.3 shows the results for the according precision recall tests of the full pipeline, varying the threshold value ϵ of the NCC-based image comparison. The different curves correspond to varying the number of nearest neighbors. Within the accuracy of this test, the parameter does not seem to affect the results and 10 nearest neighbors appear to be sufficient. However, the test scene is still rather small; in very large scenes, an increase might still provide an advantage. Experimenting with the full Hannover data set, we found that 30 nearest neighbors are usually sufficient. A full assessment would require large scale annotations on large quantities of data, which we have to leave for future work. We have also varied the other descriptor parameters (not shown here) and confirmed the results from the previous test.

In terms of absolute recognition performance, the results are quite encouraging. When taking the transitive closure of the matches into account, the recognition rate rapidly approaches 100% at small false-positive rates. For this further test, we read off an NCC threshold value of 0.5 that gives almost 100% accuracy at minimal false positive rates, and use it for all further experiments.



(a) influence of different number of histograms



(b) influence of dimensionality reduction and orientation binning

Figure 4.2: Precision recall curves for the descriptor test. Annotations: **H** = number of spatial histograms, **D** = dim. after PCA, **B** = number of orientation bins. The y-axis is the true positive rate, averaged over all annotations, and the x-axis is the false-positive rate, in percentage of all descriptors.

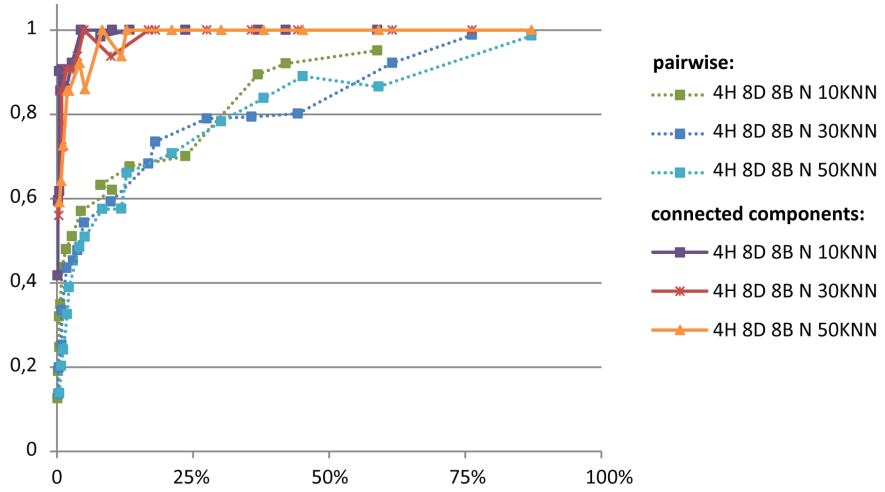


Figure 4.3: Precision recall curves for the full pipeline. Annotations: **H** = number of spatial histograms, **D** = dim. after PCA, **B** = number of orientation bins, **KNN** = number of nearest neighbors in clustering. y-axis: true positive rate, x-axis: false-positive rate. The lower three curves show the pairwise recognition rate, the upper three make the more realistic assumption of extracting the connected components of the transitive closure.

Scalability: Figure 4.4 examines the scaling behavior of our method. We replicate the museum scene on a regular grid to obtain up to 500GB of input data. Both, the overall computation time (from line feature extraction up to including clustering) and the clustering step itself scale almost linearly with the scene size. When going from the smallest to the next bigger scene, the increase is even sublinear because of an uneven load balancing in the parallel line feature computation that negatively affects small workloads. The absolute running times are also quite satisfactory: Computing symmetries on a 490GB input scene took overall below 17 hours, and the single Hannover scan collection required just 66 minutes. About two thirds of the time are spend on computing line features and feature points, one third is spend in the actual clustering-based symmetry detection. Table 4.1 summarizes the statistics and timings of all of the test scenes.

Immediate results: For illustration, we show the immediate results of the detection pipeline for the "old town hall" scan in Figure 4.5, demonstrating how the line features capture important geometric aspects, robust to partial and noisy data, and how the descriptors serve as a prefilter for the geometric validation. Please note that descriptors are shown in a three dimensional RGB projection; the actually employed descriptors are 8-dimensional and thus more discriminative than they appear in the projection.

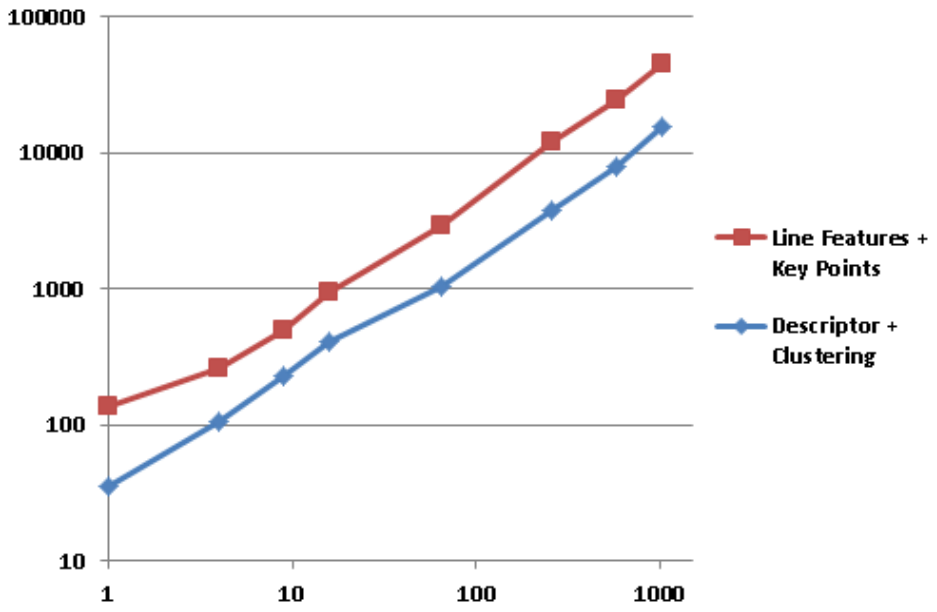
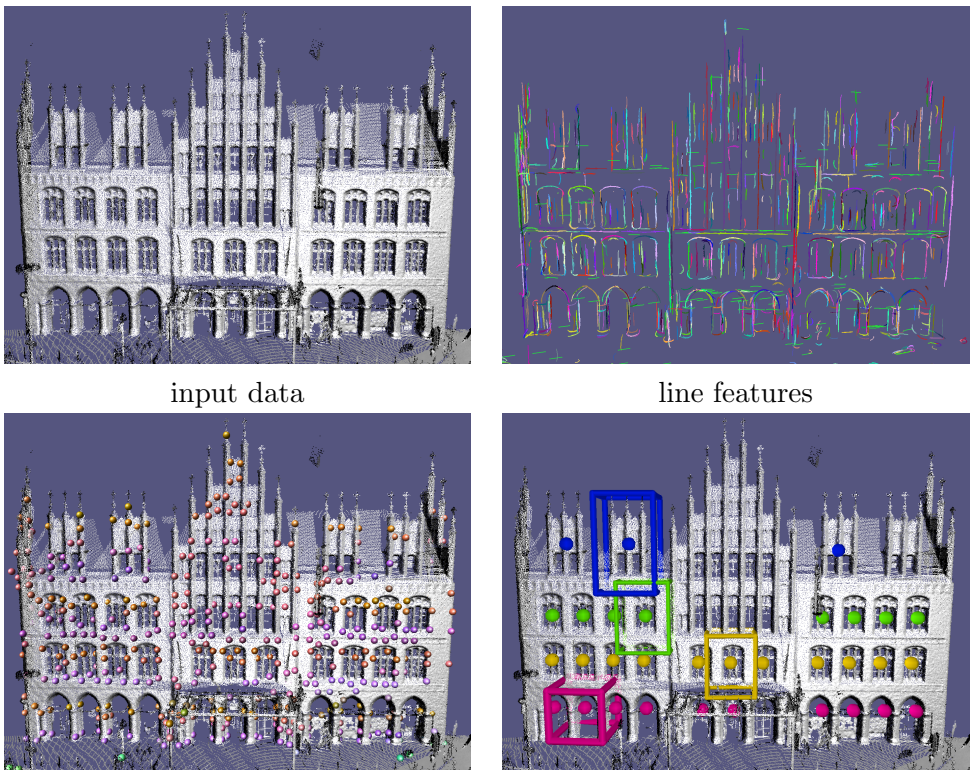


Figure 4.4: Scaling behavior with increasing scene size. The x-axis depicts the number of instances of the Museum scene (15.3M points each, corresponding to 488MB). The y-axis depicts elapsed time in seconds. Please note that both axes are logarithmically scaled.

Model	#Instances	Size	#Points	Line Features + Key Points	Descriptors + Clustering
Museum	1	0.5GB	15.3M	2m 18s	35s
	4	2GB	61.3M	4m 20s	1m 45s
	9	4GB	138.0M	8m 22s	3m 46s
	16	7GB	245.4M	15m 43s	6m 43s
	64	30GB	981.6M	48m 52s	17m 32s
	256	122GB	3.9B	3h 23m 27s	1h 2m 19s
	576	276GB	8.8B	6h 49m 18s	2h 12m 17s
	1024	490GB	15.7B	12h 29m 2s	4h 15m 47s
Hannover	1	14GB	463.4M	43m 57s	23m 12s
	9	128GB	4.2B	5h 19m 4s	3h 50m 14s

Table 4.1: Runtimes of the two algorithm stages for different models.



input data | line features

descriptors projected to 3D (RGB) via PCA | dynamic query

Figure 4.5: Visualization of the symmetry detection pipeline stages. The raw point cloud is first reduced to line features (colors indicate local line segments with a common tangential direction), from which descriptors are derived (visualized here by an RGB embedding). After clustering, a dynamic query extracts area-based symmetries.

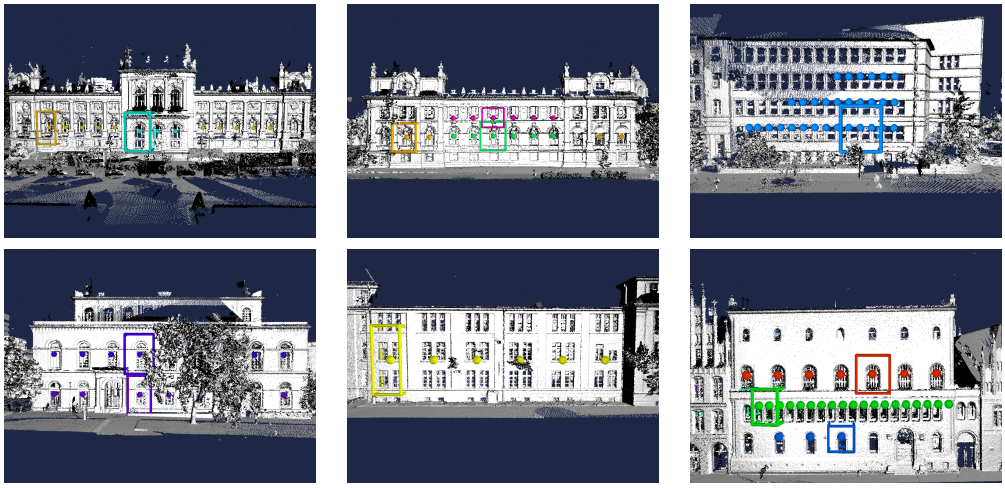


Figure 4.6: Different symmetries found across the city of Hannover. Please note that this test uses a single parameter set for the whole data set. Detection and query are performed globally, on the whole city simultaneously (see the accompanying video for details).

4.2 Full Hannover Data Set

In Figure 4.6 we show a set of example results, all obtained from a single symmetry detection pass on the Hannover scan collection. For clarity, we show a selection of dynamic queries, i.e., not all symmetry information is displayed simultaneously in the results. We indicate the selected symmetries by a colored sphere where each user query has an individual random color. The user-marked bounding box is shown in the same color accordingly. Please note that the search for both clustering and dynamic queries is performed globally, on the whole data set. The computation of a dynamic query required a few seconds each. Our results are robust even in the presence of substantial noise, clutter, and missing data (to get an impression of the data quality we refer to the accompanying video).

Comparison to Bokeloh et al. 2009 [8]: A quantitative comparison is difficult because the previous technique computes symmetric parts by greedy region growing. Depending on parameter settings, this will yield quite different results in terms of detected symmetries. On the contrary, our method precomputes the richer set of all point-wise symmetry cliques for all feature points. However, we then use these for a dynamic query with specified area, an information that the previous technique cannot rely on. Nevertheless, it is worth attempting a qualitative comparison as shown in Figure 4.7.

Bokeloh et al. 2009 scalable detection

Figure 4.7: Comparison to Bokeloh et al. 2009

For the old town hall scene (top row), we can observe the differences over greedy region growing. The previous method detects some of the windows but omits other details in favor of a global reflective symmetry of the whole facade (which our approach does not recognize as we omit reflections from the set $\mathcal{T}$). However, our method collects all symmetry information for the details and reports them when queried, with only one top piece missing. Unlike the previous method, we can also differentiate the different window variants correctly. For the new town hall (bottom row), different parameter settings for the region growing lead to a more favorable, fine grained decomposition of the facade. Our technique cannot capture very small details individually (such as the small windows in the top row), but otherwise leads to comparable results. Please note that our method processed the *entire* data at once with one set of parameters; in [8] parameters were set to each data set individually. Furthermore, our new method is less parameter dependent, only the sensitivity and scale of the line-feature detection need to be adapted to the data set, further parameters are kept constant.

4.3 Discussion

Recognition results: The recognition results of our method for architectural scenes are comparable with the previous state-of-the-art in symmetry detection that are restricted to small data sets. However, our method is restricted as it does not directly output dense correspondences but only matches at feature points that sample the model less densely. Dense correspondences still have to be reconstructed from this information, as done for example in the presented dynamic query algorithm. Precomputing such information as it was done in previous work is not practical for large scenes. Efficient algorithms and data structures for computing and storing such overlapping dense correspondence information is still subject to future work. Nevertheless, we would argue that the prime problem is finding symmetric instances in the first place, which is what this paper addresses.

Scalability: Our method is currently designed for handling large but not extremely large scenes. A key limitation is that the line features and samples need to fit into main memory. Observing an empirical compression factor of about 170, this would in principle allow us to handle scenes up 1-10TB using affordable 6-60 GB of main memory. However, the computation of the line features then becomes a bottleneck. In addition, data partitioning uses up twice the amount of time, but this could be optimized by removing the multi-resolution rendering capabilities, which currently dominates the preprocessing times [53]. Another concern is the effectiveness of the descriptor in large scenes with a large variety of geometry. While we were not able to examine this on Terabyte-sized scenes due to the lack of publicly available data, our experience with the Hannover test set was rather positive. Despite significant geometric variety and severe noise, missing data and clutter, recognition results and clustering costs were satisfactory, with results comparable to the state-of-the-art in small scenes.

Further limitations: Our method is limited by construction to detect symmetries that share a fixed upward orientation, and does not consider reflections. In the context of city scanning this is acceptable since common architectural building blocks meet this requirement. However, this might be an issue for other data sets, such as large scale data from medical or cellular imaging in biology. The main challenge for obtaining full rotational invariance is reformulating the descriptor for full coordinate frame invariance. Another limitation of our method is that it depends on the existence of line features. Especially small scale details often tend to produce insufficient line features which results in undetected symmetries. For example, in Figure 4.6, top-right image, the decrease of sampling density removed too many line features for successful detection.

5 Conclusions and Future Work

We presented a new symmetry detection method designed for large-scale point cloud data. The key idea is to design a feature space in which nearby points have symmetric geometry with high likelihood and then perform clustering in that space, which only involves local comparisons such that the method is able to scale to large data sets. We have introduced a new descriptor for 3D point clouds that is optimized for low dimensionality and fast computation. Furthermore, we have designed a very fast approximate geometry matching scheme that is optimized for architectural objects and two orders of magnitude faster than previous line-feature alignment, and thereby four orders of magnitude faster than plain pairwise ICP of the original data points. In combination, we obtain a symmetry detection system that can handle very large point sets on a single PC. We have demonstrated symmetry detection in scenes up to 490GB of raw input data, which is about two orders of magnitude larger than the results of previous work.

In future work, we would like to examine scalability to even larger data sets, beyond several TB in size. This would require a distributed implementation on a cluster, and a CUDA implementation could help in further improving the absolute throughput. Existing distributed algorithms for out-of-core clustering of image data as described in [27] could serve as an inspiration for a generalization in this direction. In addition to that, as discussed above, it would also be interesting to find a compact encoding of actual matching geometry, and remove the limitations of only matching feature points.

Bibliography

[1] A. Adams, N. Gelfand, and K. Pulli. Viewfinder alignment. *Computer Graphics Forum*, 27(2):597–606, 2008.

[2] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *International Conference on Computer Vision*, 2009.

[3] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[4] D. Anguelov, P. Srinivasan, H.-C. Pang, D. Koller, S. Thrun, and J. Davis. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *NIPS*, 2004.

[5] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45:891–923, 1998.

[6] A. Berner, M. Bokeloh, M. Wand, A. Schilling, and H.-P. Seidel. A graph-based approach to symmetry detection. In *Proc. Symp. Point-Based Graphics 2008*, 2008.

[7] A. Berner, M. Wand, N. Mitra, D. Mewes, and H.-P. Seidel. Shape analysis with subspace symmetries. 30(2), April 2011.

[8] M. Bokeloh, A. Berner, M. Wand, H.-P. Seidel, and A. Schilling. Symmetry detection using line features. *Computer Graphics Forum*, 28(2), 2009.

[9] M. Bokeloh, M. Wand, V. Koltun, and H.-P. Seidel. Pattern-aware shape deformation using sliding dockers. *ACM Transactions on Graphics (Proc. Siggraph Asia 2011)*, 30(5):to appear., 2011.

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

[11] C. Brenner. Hannover city scan database. http://www.ikg.uni-hannover.de/index.php?id=413, 2007.

[12] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Efficient computation of isometry-invariant distances between surfaces. *SIAM J. Sci. Comput.*, 28(5):1812–1836, 2006.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, June 2005.

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

[15] J.-M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building rome on a cloudless day. In *European Conference on Computer Vision*, 2010.

[16] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *Intl. Workshop on Automatic Face and Gesture-Recognition*, pages 296–301. IEEE Computer Society, 1995.

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

[18] R. Gal, A. Shamir, T. Hassner, M. Pauly, and D. Cohen-Or. Surface reconstruction using local shape priors. In *Proc. Symp. Geometry Processing*, 2007.

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

[20] N. Gelfand and L. Guibas. Shape segmentation using local slippage analysis. In *Proc. Symp. Geometry Processing*, 2004.

[21] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Proc. Symp. Geometry Processing*, pages 197–206, 2005.

[22] J. Goldstein, J. C. Platt, and C. J. C. Burges. Redundant bit vectors for quickly searching high-dimensional regions. *Deterministic and Statistical Methods in Machine Learning*, 3635:137–158, 2005.

[23] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:433–449, 1999.

[24] M. Kazhdan, B. Chazelle, D. Dobkin, T. Funkhouser, and S. Rusinkiewicz. A reflective symmetry descriptor for 3d models. *Algorithmica*, 38(1):201–225, 2003.

[25] J. Lin, D. Cohen-Or, H. R. Zhang, C. Liang, A. Sharf, O. Deussen, and B. Chen. Structure-preserving retargeting of irregular 3d architecture. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2011)*, 30(6), dec 2011.

[26] Y. Lipman, X. Chen, I. Daubechies, and T. Funkhouser. Symmetry factored embedding and distance. *ACM Transactions on Graphics (SIGGRAPH 2010)*, July 2010.

[27] T. Liu, C. Rosenberg, and H. A. Rowley. Clustering billions of images with large scale nearest neighbor search. In *Proceedings of the Eighth IEEE Workshop on Applications of Computer Vision*, WACV '07, Washington, DC, USA, 2007. IEEE Computer Society.

[28] Y. Liu, R. Collins, and Y. Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):354–371, March 2004.

[29] D. Lowe. Distinctive image features from scale-invariant keypoints. In *Int. J. Computer Vision*, volume 20, pages 91–110, 2003.

[30] G. Loy and J. Eklundh. Detecting symmetry and symmetric constellations of features. In *ECCV*, pages 508–521, 2006.

[31] C. Maes, T. Fabry, J. Keustermans, D. Smeets, P. Suetens, and D. Vandermeulen. Feature detection on 3d face surfaces for pose normalisation and recognition. In *Biometrics: Theory Applications and Systems (BTAS)*, 2010.

[32] A. Martinet, C. Soler, N. Holzschuch, and F. Sillion. Accurate detection of symmetries in 3d shapes. *ACM Trans. on Graphics*, 25(2):439 – 464, 2006.

[33] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[34] N. J. Mitra, N. Gelfand, H. Pottmann, and L. Guibas. Registration of point cloud data from a geometric optimization perspective. In *Symp. Geometry Processing*, 2004.

[35] N. J. Mitra, L. Guibas, and M. Pauly. Symmetrization. In *ACM Transactions on Graphics*, volume 26, 2007.

[36] N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Trans. Graph.*, 25(3):560–568, 2006.

[37] N. J. Mitra and M. Pauly. Symmetry for architectural design. In *Advances in Architectural Geometry*, pages 13–16, 2008.

[38] N. J. Mitra, M. Pauly, M. Wand, and D. Ceylan. Symmetry in 3d geometry: Extraction and applications. EUROGRAPHICS State-of-the-art Report (STAR), 2012.

[39] N. J. Mitra, Y.-L. Yang, D.-M. Yan, W. Li, and M. Agrawala. Illustrating how mechanical assemblies work. *ACM Transactions on Graphics*, 29(3), 2010.

[40] Y. Ohtake, A. Belyaev, and H.-P. Seidel. Ridge-valley lines on meshes via implicit surface fitting. In *SIGGRAPH*, pages 609–612, 2004.

[41] M. Ovsjanikov, J. Sun, and L. Guibas. Global intrinsic symmetries of shapes. In *Eurographics Symposium on Geometry Processing (SGP)*, 2008.

[42] M. Park, R. Collins, and Y. Liu. Deformed lattice discovery via efficient mean-shift belief propagation. *ECCV 2008*, pages 474–485, 2008.

[43] M. Parky, S. Leey, P.-C. Cheny, S. Kashyap, A. A. Butty, and Y. Liu. Performance evaluation of state-of-the-art discrete symmetry detection algorithms. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2008.

[44] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled models. In *Proc. Eurographics*, 2003.

[45] M. Pauly, N. Mitra, J. Giesen, M. Gross, and L. J. Guibas. Example-based 3d scan completion. In *Proc. Symp. Geometry Processing*, 2005.

[46] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. Guibas. Discovering structural regularity in 3D geometry. *ACM Trans. Graph.*, 27(3), 2008.

[47] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser. A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 25(3), July 2006.

[48] D. Raviv, A. Bronstein, M. Bronstein, and R. Kimmel. Symmetries of non-rigid shapes. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–7, Oct. 2007.

[49] R. T. C. Seungkyu Lee and Y. Liu. Rotation symmetry group detection via frequency analysis of frieze-expansions. In *Computer Vision and Pattern Recognition (CVPR)*, June 2008.

[50] P. Simari, E. Kalogerakis, and K. Singh. Folding meshes: hierarchical mesh segmentation based on planar symmetry. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 111–119, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[51] M. Sunkel, S. Jansen, M. Wand, E. Eisemann, and H.-P. Seidel. Learning line features in 3d geometry. 30(2), April 2011.

[52] S. Thrun and B. Wegbreit. Shape from symmetry. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1824–1831, Washington, DC, USA, 2005. IEEE Computer Society.

[53] M. Wand, A. Berner, M. Bokeloh, P. Jenke, A. Fleck, M. Hoffmann, B. Maier, D. Staneker, A. Schilling, and H.-P. Seidel. Special section: Point-based graphics: Processing and interactive editing of huge point clouds from 3d scanners. *Comput. Graph.*, 32:204–220, April 2008.

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

[55] H. Zhang, A. Sheffer, D. Cohen-Or, Q. Zhou, O. van Kaick, and A. Tagliasacchi. Deformation-driven shape correspondence. *Comput. Graph. Forum*, 27(5):1431–1439, 2008.

[56] Q. Zheng, A. Sharf, G. Wan, Y. Li, N. J. Mitra, D. Cohen-Or, and B. Chen. Non-local scan consolidation for 3d urban scenes. *ACM Transactions on Graphics*, 29(3), 2010.

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

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

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

D-66123 Saarbrücken

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

MPI-I-2012-RG1-001	M. Suda, C. Weidenbach	Labelled superposition for PLTL
MPI-I-2011-5-002	B. Taneva, M. Kacimi, G. Weikum	Finding images of rare and ambiguous entities
MPI-I-2011-5-001	A. Anand, S. Bedathur, K. Berberich, R. Schenkel	Temporal index sharding for space-time efficiency in archive search
MPI-I-2011-4-005	A. Berner, O. Burghard, M. Wand, N.J. Mitra, R. Klein, H. Seidel	A morphable part model for shape manipulation
MPI-I-2011-4-001	M. Granados, J. Tompkin, K. In Kim, O. Grau, J. Kautz, C. Theobalt	How not to be seen inpainting dynamic objects in crowded scenes
MPI-I-2010-RG1-001	M. Suda, C. Weidenbach, P. Wischnewski	On the saturation of YAGO
MPI-I-2010-5-008	S. Elbassuoni, M. Ramanath, G. Weikum	Query relaxation for entity-relationship search
MPI-I-2010-5-007	J. Hoffart, F.M. Suchanek, K. Berberich, G. Weikum	YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia
MPI-I-2010-5-006	A. Broschart, R. Schenkel	Real-time text queries with tunable term pair indexes
MPI-I-2010-5-005	S. Seufert, S. Bedathur, J. Mestre, G. Weikum	Bonsai: Growing Interesting Small Trees
MPI-I-2010-5-004	N. Preda, F. Suchanek, W. Yuan, G. Weikum	Query evaluation with asymmetric web services
MPI-I-2010-5-003	A. Anand, S. Bedathur, K. Berberich, R. Schenkel	Efficient temporal keyword queries over versioned text
MPI-I-2010-5-002	M. Theobald, M. Sozio, F. Suchanek, N. Nakashole	URDF: Efficient Reasoning in Uncertain RDF Knowledge Bases with Soft and Hard Rules
MPI-I-2010-5-001	K. Berberich, S. Bedathur, O. Alonso, G. Weikum	A language modeling approach for temporal information needs
MPI-I-2010-1-001	C. Huang, T. Kavitha	Maximum cfardinality popular matchings in strict two-sided preference lists
MPI-I-2009-RG1-005	M. Horbach, C. Weidenbach	Superposition for fixed domains
MPI-I-2009-RG1-004	M. Horbach, C. Weidenbach	Decidability results for saturation-based model building
MPI-I-2009-RG1-002	P. Wischnewski, C. Weidenbach	Contextual rewriting
MPI-I-2009-RG1-001	M. Horbach, C. Weidenbach	Deciding the inductive validity of $\forall\exists^*$ queries
MPI-I-2009-5-007	G. Kasneci, G. Weikum, S. Elbassuoni	MING: Mining Informative Entity-Relationship Subgraphs
MPI-I-2009-5-006	S. Bedathur, K. Berberich, J. Dittrich, N. Mamoulis, G. Weikum	Scalable phrase mining for ad-hoc text analytics
MPI-I-2009-5-005	G. de Melo, G. Weikum	Towards a Universal Wordnet by learning from combined evidenc
MPI-I-2009-5-004	N. Preda, F.M. Suchanek, G. Kasneci, T. Neumann, G. Weikum	Coupling knowledge bases and web services for active knowledge
MPI-I-2009-5-003	T. Neumann, G. Weikum	The RDF-3X engine for scalable management of RDF data
MPI-I-2009-5-002	M. Ramanath, K.S. Kumar, G. Ifrim	Generating concise and readable summaries of XML documents

MPI-I-2009-4-006	C. Stoll	Optical reconstruction of detailed animatable human body models
MPI-I-2009-4-005	A. Berner, M. Bokeloh, M. Wand, A. Schilling, H. Seidel	Generalized intrinsic symmetry detection
MPI-I-2009-4-004	V. Havran, J. Zajac, J. Drahokoupil, H. Seidel	MPI Informatics building model as data for your research
MPI-I-2009-4-003	M. Fuchs, T. Chen, O. Wang, R. Raskar, H.P.A. Lensch, H. Seidel	A shaped temporal filter camera
MPI-I-2009-4-002	A. Tevs, M. Wand, I. Ihrke, H. Seidel	A Bayesian approach to manifold topology reconstruction
MPI-I-2009-4-001	M.B. Hullin, B. Ajdin, J. Hanika, H. Seidel, J. Kautz, H.P.A. Lensch	Acquisition and analysis of bispectral bidirectional reflectance distribution functions
MPI-I-2008-RG1-001	A. Fietzke, C. Weidenbach	Labelled splitting
MPI-I-2008-5-004	F. Suchanek, M. Sozio, G. Weikum	SOFI: a self-organizing framework for information extraction
MPI-I-2008-5-003	G. de Melo, F.M. Suchanek, A. Pease	Integrating Yago into the suggested upper merged ontology
MPI-I-2008-5-002	T. Neumann, G. Moerkotte	Single phase construction of optimal DAG-structured QEPs
MPI-I-2008-5-001	G. Kasneci, M. Ramanath, M. Sozio, F.M. Suchanek, G. Weikum	STAR: Steiner tree approximation in relationship-graphs
MPI-I-2008-4-003	T. Schultz, H. Theisel, H. Seidel	Crease surfaces: from theory to extraction and application to diffusion tensor MRI
MPI-I-2008-4-002	D. Wang, A. Belyaev, W. Saleem, H. Seidel	Estimating complexity of 3D shapes using view similarity
MPI-I-2008-1-001	D. Ajwani, I. Malinger, U. Meyer, S. Toledo	Characterizing the performance of Flash memory storage devices and its impact on algorithm design
MPI-I-2007-RG1-002	T. Hillenbrand, C. Weidenbach	Superposition for finite domains
MPI-I-2007-5-003	F.M. Suchanek, G. Kasneci, G. Weikum	Yago : a large ontology from Wikipedia and WordNet
MPI-I-2007-5-002	K. Berberich, S. Bedathur, T. Neumann, G. Weikum	A time machine for text search
MPI-I-2007-5-001	G. Kasneci, F.M. Suchanek, G. Ifrim, M. Ramanath, G. Weikum	NAGA: searching and ranking knowledge
MPI-I-2007-4-008	J. Gall, T. Brox, B. Rosenhahn, H. Seidel	Global stochastic optimization for robust and accurate human motion capture
MPI-I-2007-4-007	R. Herzog, V. Havran, K. Myszkowski, H. Seidel	Global illumination using photon ray splatting
MPI-I-2007-4-006	C. Dyken, G. Ziegler, C. Theobalt, H. Seidel	GPU marching cubes on shader model 3.0 and 4.0
MPI-I-2007-4-005	T. Schultz, J. Weickert, H. Seidel	A higher-order structure tensor
MPI-I-2007-4-004	C. Stoll, E. de Aguiar, C. Theobalt, H. Seidel	A volumetric approach to interactive shape editing
MPI-I-2007-4-003	R. Bargmann, V. Blanz, H. Seidel	A nonlinear viseme model for triphone-based speech synthesis
MPI-I-2007-4-002	T. Langer, H. Seidel	Construction of smooth maps with mean value coordinates
MPI-I-2007-4-001	J. Gall, B. Rosenhahn, H. Seidel	Clustered stochastic optimization for object recognition and pose estimation
MPI-I-2007-2-001	A. Podelski, S. Wagner	A method and a tool for automatic veriication of region stability for hybrid systems
MPI-I-2007-1-003	A. Gidenstam, M. Papatriantafilou	LFthreads: a lock-free thread library
MPI-I-2007-1-002	E. Althaus, S. Canzar	A Lagrangian relaxation approach for the multiple sequence alignment problem
MPI-I-2007-1-001	E. Berberich, L. Kettner	Linear-time reordering in a sweep-line algorithm for algebraic curves intersecting in a common point