Eurographics Symposium on Point-Based Graphics (2004)
M. Alexa, S. Rusinkiewicz (Editors)

Post-processing of Scanned 3D Surface Data

T. Weyrich[1], M. Pauly[2], R. Keiser[1], S. Heinzle[1], S. Scandella[1] and M. Gross[1]

[1] Department of Computer Science, ETH Zürich, CH-8092 Zürich, Switzerland
[2] Computer Science Department, Stanford University, USA

Abstract

3D shape acquisition has become a major tool for creating digital 3D surface data in a variety of application fields. Despite the steady increase in accuracy, most available scanning techniques cause severe scanning artifacts such as noise, outliers, holes, or ghost geometry. To apply sophisticated modeling operations on these data sets, substantial post-processing is usually required. In this paper, we address a variety of scanning artifacts that are created by common optical scanners and provide a comprehensive set of user-guided tools to process corrupted data sets. These include an eraser tool, low-pass filters for noise removal, a set of outlier detection methods, and various up-sampling and hole-filling tools. These techniques can be applied early in the content creation pipeline. Therefore, all our tools are implemented to operate directly on the acquired point cloud. We also emphasize the need for extensive user control and an efficient visual feedback loop. The effectiveness of our scan cleaning tools is demonstrated on various models acquired with commercial laser-range scanners and low-cost structured light scanners.

Categories and Subject Descriptors (according to ACM CCS): I.3.5. [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

With growing demand for realism in computer graphics and interactive techniques, we experience a steady increase in the geometric complexity of digital 3D surface models. Ab initio design of such shapes thus becomes increasingly time consuming and expensive. Most designers therefore rely on 3D scanning devices to acquire complex digital models from real-world objects. Accurate 3D acquisition also plays an important role in reverse engineering, rapid prototyping, biomedicine, architecture, cultural heritage acquisition, or entertainment industry.

This diversity in application fields is reflected in a great variety of 3D imaging techniques: CT and MRI scanners are widely used in medical and engineering applications to acquire volumetric representations of real-world objects. Optical devices, such as laser-range scanners or structured light scanners, are primarily employed for surface and appearance acquisition.

This latter class of scanning devices typically produces a dense set of surface points, where each point samples a 3D position and possible additional attributes such as normal information, color, or material properties. Depending on the specific acquisition method, a number of scanning artifacts can occur as illustrated in Figure 1:

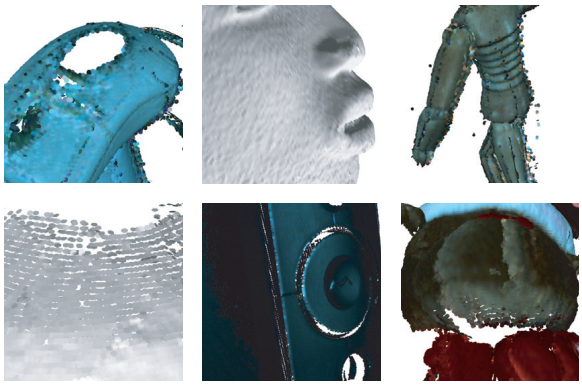


Figure 1: *Typical artifacts of raw scanner data.* Top Row: *Holes due to sensor restrictions, noise, outliers.* Bottom Row: *Low sampling density due to gracing sensor views, low sampling density at delicate surface details, and holes due to critical reflectance properties.*

- Physical limitations of the sensor lead to noise in the acquired data set. Sample points can also be corrupted by

© The Eurographics Association 2004.

quantization or motion artifacts. The latter occur when the scanned object moves during the acquisition process, a common problem when scanning humans or animals.
- Multiple reflections and heavy noise can produce off-surface points (outliers).
- Holes and under-sampling in the model surface occur due to occlusion, critical reflectance properties, constraints in the scanning path, or limited sensor resolution.
- Many scanners tend to create ghost geometry when the scanned object is textured.

The raw point cloud data produced by the scanner thus needs to be processed before subsequent modeling operations can be performed. Commercial scanners are usually equipped with rudimentary scan cleaning software that uses built-in heuristics for outlier removal and noise reduction. These are often difficult to control as they are optimized for the specific scanner configuration.

More sophisticated data processing can only be applied by exporting the acquired surface model from the proprietary scanner software, typically in the form of a triangle mesh. However, if the above mentioned data imperfections have not been successfully removed from the data set, the meshing process itself is fragile and can even introduce further artifacts. We thus argue that post-processing of scanned data should be performed directly on the acquired point cloud, before sophisticated surface reconstruction algorithms or advanced modeling operations are applied.

To this end, we propose a purely point-based scan cleaning toolbox, consisting of a selection of user-guided tools that address the different scanning artifacts mentioned above. These include an eraser tool, low-pass filters for noise removal, a set of outlier detection methods, and various resampling and hole-filling tools.

Since many scan artifacts are strongly coupled, these tools should be applied in an interleaved fashion. Identification of artifacts is difficult and often requires human interpretation. Therefore, user guidance is a necessary prerequisite to achieve optimal results. We specifically designed our algorithms to support rapid feedback during an interactive scan cleaning session. This allows the user to interactively adjust tool parameters, such as outlier thresholds or filter transfer functions.

We have integrated our scan cleaning toolbox as a plug-in into Pointshop3D, an open-source 3D editing tool for point-sampled surfaces [24, 19]. In combination with a 3D scanning front end, our plug-in bridges the gap between 3D acquisition and high-level shape and appearance modeling, thus providing in a single application a complete point-based content creation pipeline.

2. Related Work

Noise and outliers can be removed by applying a spatial depth-pass filter to the 3D point data [17]. Alternatively, noise can implicitly be handled during a surface reconstruction stage. [1, 5, 13] yield a smooth surface by approximating the sample points. However, most automatic surface reconstruction algorithms fail in the presence of severe noise and outliers.

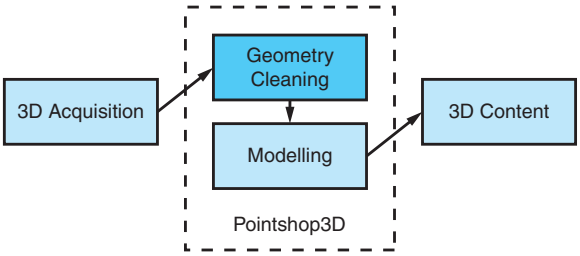


Figure 2: *Our toolbox bridges the gap between 3D acquisition and higher-level modeling. As a Pointshop3D plugin it aims at a completely point-based content creation pipeline.*

In the past, various hole-filling techniques have been proposed. These methods mostly use implicit representations to define the underlying surface. Verdera et al. [22] extend image inpainting techniques to 3D surfaces by solving anisotropic partial differential equations defined on the surface. Carr et al. [5] and Ohtake et al. [13] exploit the extrapolation properties of radial basis functions to fill regions of sparse sampling. Davis et al. [6] propose a method that applies a diffusion operator on the signed distance field of an incomplete triangle mesh.

Only little has been published on the user-guided cleaning of raw scanner output. [4] analyzes requirements of scan cleaning software and gives a short overview over existing commercial systems. Those systems usually triangulate the data at an early stage, before fitting higher-order surface representations. However, to the best of our knowledge, a system for 3D scan cleaning directly working on the point cloud data has not been published yet.

3. Overview

The central motivation of our toolbox has been to open up modeling techniques to be used for the cleaning of raw scan data. Our modeling tools make extensive use of basic techniques (Section 4), which are well-known in point graphics community or adapted from triangle based graphics, respectively.

Section 5 suggests a set of tools and discusses the underlying design criteria. It is explained how the basic techniques are extended and combined to realize the different tools. The integration of the tools in a common user interface is presented.

Section 6 demonstrates some exemplary steps of the cleaning procedure. It shows how the single tools interact when cleaning raw scan data.

© The Eurographics Association 2004.

4. Basic Techniques

The presented toolbox internally utilizes a set of basic geometric modeling techniques. This section describes the respective techniques and explains their adaption to point clouds.

4.1. Search Data Structures

Dealing with point clouds, we do not have any explicit connectivity information. This means that all computations are based on spatial proximity between point samples instead of geodesic proximity between mesh vertices. In this section we present two data structures for fast nearest neighbor searches and range queries.

A very well known search data structure is the k-d tree (e.g. [3, 7, 2]). A k-d tree can be searched efficiently in $O(\log n)$, while it takes time $O(n \log n)$ to build it. Therefore, and because it is costly to maintain a k-d tree after an insertion, deletion or displacement of points, it is suitable for static data only. If the same point is queried more than once, it might be useful to cache the neighbors. In this case, a nearest-neighbor graph is built, storing the nearest neighbors for each point.

For querying dynamic data we use a hash data structure similar to [20]. The coordinates of an arbitrary point in space are mapped to a cell. If the cell size is chosen smaller or equal than the maximal query range, all points within this range can be found by searching the adjacent cells to a query point, i.e. 27 cells have to be queried. Note that also k-nearest neighbor queries can be performed efficiently if a maximal range can be given. However, while insertion of a point can be done in $O(1)$, querying takes $O(q)$, where q is the maximum number of points in a cell. In practice, with a sufficient number of cells q will be small.

4.2. MLS Projection

To compute a smooth surface that approximates a set of scattered data points, Levin [11] introduced a projection operator based on Moving Least Squares (MLS) optimization. Using this projection procedure, Alexa et al. [1] presented a high quality rendering algorithm for point set surfaces. Because the MLS method is crucial for the following algorithms, we will briefly review it.

Let P be an unstructured set of sample points. The MLS projection takes a point $\boldsymbol{x}$ in space and projects it onto a polynomial that locally approximates the underlying surface in the vicinity of $\boldsymbol{x}$. This polynomial is computed by first fitting a reference plane H using weighted least squares optimization. The reference plane provides a local parameterization of the sample points, which is used in a second least squares fit to compute a bivariate polynomial approximation.

Both, the computation of the reference plane and the polynomial use a radially symmetric Gaussian weight function $\omega_i = \mathrm{e}^{-\|\boldsymbol{x}_r - \boldsymbol{p}_i\|^2/h^2}$, where $\boldsymbol{x}_r$ is the projected point of $\boldsymbol{x}$ onto H and h is a scaling factor. Since ω_i drops quickly with increasing distance, the least squares optimization is typically applied in a local neighborhood around the point of interest. The scaling factor h can either be a global constant or proportional to local sample spacing, estimate from a k-neighborhood as described in [16]. More details on the MLS method can be found in [11] and [1].

4.3. Point Relaxation

In [21], Turk uses particle simulation for resampling polygonal surfaces. Pauly et al. [16] adapted this method to point-sampled surfaces.

To achieve a uniform distribution of the particles, we let neighbored particles repel each other. Every particle $\boldsymbol{p}$ exerts a force $\boldsymbol{f}_i(\boldsymbol{p})$ on its neighbored particles $\boldsymbol{p}_i$. The summation of all forces that act on a particle gives the resulting force. Finally, the new positions of the particles are computed by explicit Euler integration.

We use the same repulsion force f as in [21, 16]:

$$\boldsymbol{f}_i(\boldsymbol{p}) = k(r - \|\boldsymbol{p} - \boldsymbol{p}_i\|)\frac{\boldsymbol{p}_i - \boldsymbol{p}}{\|\boldsymbol{p}_i - \boldsymbol{p}\|}, \qquad (1)$$

where k is a force constant and r is the repulsion radius. For finding the nearest neighbors within the radius r we use the hash data structure described in Section 4.1.

After each iteration, the particles are projected back onto the surface by applying the MLS projection described above. In our case, the particle simulation is performed locally for a selected region. To ensure that the selected surfels keep within this region, we compute for each selected surfel its n nearest neighbors and add the neighbors which are not selected to a list. While these surfels repel the selected surfels, their positions are fixed.

5. Tools

We built a set of tools allowing for interactive control of the presented techniques. The toolbox was designed to allow the removal of typical scan artifacts, as depicted in Figure 1. In order to support an efficient scan cleaning process, we pursued three design goals:

Predictability In order to allow a rapid workflow, it is important that each tool's effect is predictable under most circumstances. That is, if the user chooses a tool for a certain purpose, the outcome should meet the user's expectations.

Controllability The range of application must be well-controllable. Where possible, each tool should provide a set of parameters to tune its behaviour.

Intuitive Handling The tools should rest upon intuitive editing metaphors. Any parameters should correspond to meaningful traits.

© The Eurographics Association 2004.

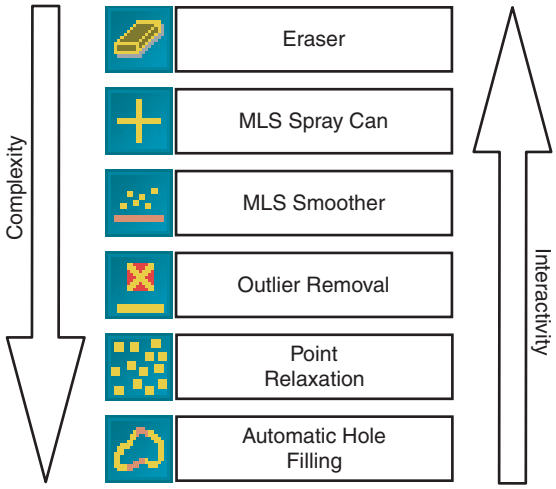


Figure 3: *The toolbox contains tools of different complexity. Higher complexity goes with less interaction.*

Following those criteria, we wanted to make the tools as powerful as possible. However, making a tool powerful usually implies the use of higher-level automatisms which are likely to fail when applied to raw scanner data. This would contradict predictability. Increasing the number of parameters to make the outcome more controllable would lead to an unintuitive handling.

We finally decided for a set of tools differing in complexity (see Figure 3). Simpler, more robust, tools allow for direct editing, especially in the presence of severe scanning artifacts. More complex and powerful tools can be applied at a later point in the scan cleaning process, when a certain sampling quality has already been achieved.

In order to address controllability, all tools provide an exhaustive set of parameters that can be set using the user interface. Each tool comes with a set of reasonable default parameters.

Most of the tools utilize a volumetric selection tool as a common interface, allowing a consistent intuitive handling. In the remainder of this section we describe the common selection mechanism and the set of basic tools.

5.1. Volumetric Selection

For most of the tools it makes sense to apply them locally. Consequently, they are defined to work on a set of selected surfels.

Pointshop3D provides a selection mechanism. However, the Pointshop3D selection tool requires a well-sampled surface and can not, e.g., select scattered points, as they frequently appear in real-world scans. We developed a volumetric brush to facilitate the selection of surfels in areas where no properly sampled surface exists. The brush, box shaped or ellipsoidal, can freely be moved in space, or alternatively follows the object surface (see Figure 4). By resizing and rotating the brush, its shape can be adapted to the local object geometry.

The brush is designed to follow the object surface even in poorly sampled regions. This is achieved by analyzing the depth values of all surfels visible around the mouse pointer. The brush's depth is set to a robust mean of the different depth values.

All tools that support the volumetric selection can be applied to the set of selected surfels. Alternatively, they are simultaneously applied to all points within the volumetric brush during navigation.

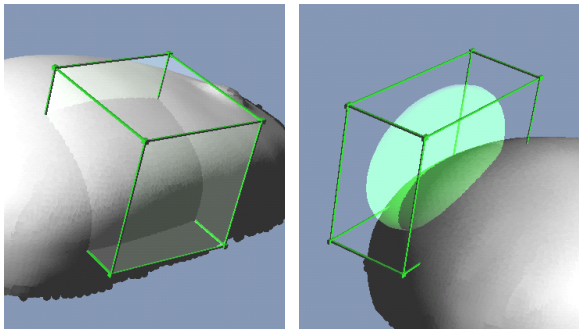
5.2. Eraser

The most primitive tool simply removes all selected points, or points within the volumetric brush, respectively. Despite its simplicity, the *eraser* is one of the most frequently employed tools.

5.3. Outlier Removal

Erroneous points outside the object surface are outliers that have to be removed. However, it is hard to specify a general criterion to detect outliers, if the real object surface is unknown. Noise further complicates the detection of outliers. In many cases, the scan quality has to be judged by the user in order to tell a noisy surface point from an outlier.

We developed an interactive tool for outlier removal incorporating the user into the outlier detection. The tool provides three outlier classification heuristics that have to be weighted by the user to obtain an appropriate classification (see Figure 5). Outliers are finally removed by applying a threshold to the resulting outlier classification.



(a) (b)

Figure 4: *The volumetric brush.* (a) *A box shaped selector, following the object surface.* (b) *An ellipsoidal selector, freely positioned in space.*

© The Eurographics Association 2004.

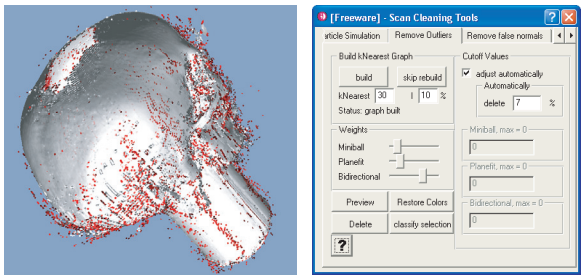


Figure 5: *Outlier classification. The three classifiers can be weighted using the depicted sliders. Probable outliers, scheduled for removal according to the resulting classification and a given threshold, are rendered in red.*

The threshold can be chosen manually. Alternatively, it is automatically set to discard a certain percentage of the points. Outlier classification can be confined to the volumetric brush.

We now present the three underlying outlier criteria. All criteria deliver an estimator $\chi(\boldsymbol{p}) \in [0,1]$ assigning the likelihood for a point sample $\boldsymbol{p}$ to be an outlier. To prevent any bias from an intermediate surface representation, all criteria are based only on analysis of $\boldsymbol{p}$'s k nearest neighbors $\mathcal{N}_{\boldsymbol{p}}$.

The respective properties of the proposed criteria will be discussed in Section 6.

5.3.1. Plane Fit Criterion

An intuitive criterion is the point's deviation from a manifold approximating its neighbors. The *plane fit criterion* considers a plane H that minimizes the squared distances to $\boldsymbol{p}$'s neighbors:

$$\min_{H} \sum_{\boldsymbol{q}\in\mathcal{N}_{\boldsymbol{p}}} dist(\boldsymbol{p},H)^2 \qquad (2)$$

(see Figure 6). Let d be the distance of $\boldsymbol{p}$ to H, and $\bar{d}$ the mean distance of points from $\mathcal{N}_{\boldsymbol{p}}$ to H. We define the plane fit criterion as

$$\chi_{\mathrm{pl}}(\boldsymbol{p}) = \frac{d}{d+\bar{d}}. \qquad (3)$$

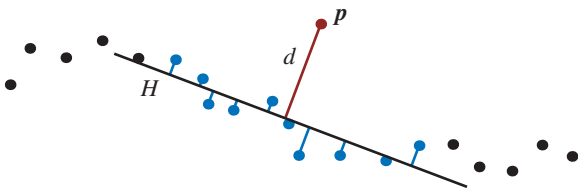


Figure 6: *The plane fit criterion compares $\boldsymbol{p}$'s distance d to a least squares plane H with the average distance of its neighbors to H. $\boldsymbol{p}$'s k-neighbors are denoted in blue.*

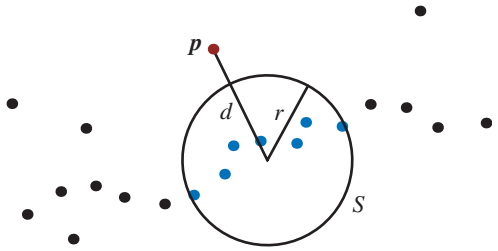


Figure 7: *The miniball criterion. A miniball S approximates the cluster of $\boldsymbol{p}$'s neighbors. The criterion compares $\boldsymbol{p}$'s distance to S with the diameter of the sphere.*

Normalization by $\bar{d}$ relates d to possible noise and surface deviations.

Instead of H, it would be possible to use higher-order approximations of $\mathcal{N}_{\boldsymbol{p}}$. We chose the plane-fit criterion to achieve a maximum of robustness.

5.3.2. Miniball Criterion

A point comparatively distant to the cluster built by its k nearest neighbors is likely to be an outlier. This observation leads to the following criterion.

For each point $\boldsymbol{p}$ consider the smallest enclosing sphere S around $\mathcal{N}_{\boldsymbol{p}}$ [23] (see Figure 7). S can be seen as an approximation of the k-nearest-neighbor cluster. Comparing $\boldsymbol{p}$'s distance d to the center of S with the sphere's diameter yields a measure for $\boldsymbol{p}$'s likelihood to be an outlier. Consequently we define the *miniball criterion* as

$$\chi_{\mathrm{mb}}(\boldsymbol{p}) = \frac{d}{d+2r/\sqrt{k}}. \qquad (4)$$

Normalization by $\sqrt{k}$ compensates for the diameter's increase with increasing number of k-neighbors at the object surface.

5.3.3. Nearest-Neighbor Reciprocity Criterion

This criterion is based on the following observation: Potential outliers draw their k-nearest neighbors from a larger vicinity than points in a well-sampled environment. In particular, a "valid" point sample $\boldsymbol{q}$ may be in the k-neighborhood of an outlier, but the outlier will most likely not be part of $\boldsymbol{q}$'s k-neighborhood.

This relationship can be expressed by means of a directed graph G of k-neighbor relationships (see Figure 8). Outliers are assumed to have a high number of uni-directional exitant edges, i.e., asymmetric neighbor relationships. Consequently the criterion considers the ratio between uni-directional and bi-directional exitant edges in G.

We define the uni-directional neighbors as $\mathcal{N}_{\mathrm{uni}}(\boldsymbol{p}) = \{\boldsymbol{q} \mid \boldsymbol{q} \in \mathcal{N}_{\boldsymbol{p}}, \boldsymbol{p} \notin \mathcal{N}_{\boldsymbol{q}}\}$, while the bi-directional neighbors

© The Eurographics Association 2004.

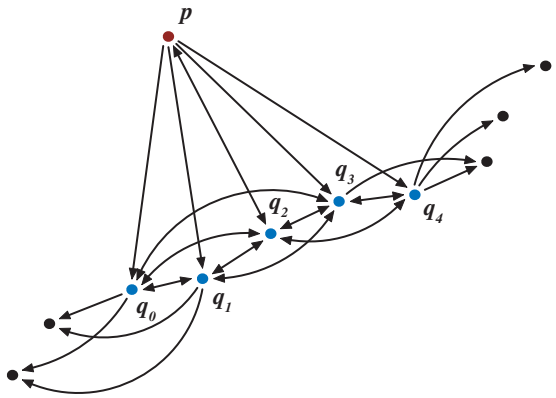


Figure 8: *Nearest-neighbor graph. Depicted are the 5-nearest-neighbor relations for* $\boldsymbol{p}$ *and its 5-neighbors* $\boldsymbol{q}_0,\ldots,\boldsymbol{q}_4$. *Note that only* $\boldsymbol{q}_2$ *shares a reciprocal neighbor relationship with* $\boldsymbol{p}$.

build a set $\mathcal{N}_{\text{bi}}(\boldsymbol{p}) = \{\boldsymbol{q} \,|\, \boldsymbol{q} \in \mathcal{N}_{\boldsymbol{p}}, \boldsymbol{p} \in \mathcal{N}_{\boldsymbol{q}}\}$. The classifier is expressed as follows:

$$\chi_{\text{bi}}(\boldsymbol{p}) = \frac{\|\mathcal{N}_{\text{uni}}(\boldsymbol{p})\|}{\|\mathcal{N}_{\text{bi}}(\boldsymbol{p})\| + \|\mathcal{N}_{\text{uni}}(\boldsymbol{p})\|} = \frac{\|\mathcal{N}_{\text{uni}}(\boldsymbol{p})\|}{k}\,. \qquad (5)$$

5.3.4. Classification

The final outlier classification is computed using weights $w_1,\ldots,w_3$, $\sum_i w_i = 1$, interactively defined by the user:

$$\chi(\boldsymbol{p}) = w_1\chi_{\text{pl}}(\boldsymbol{p}) + w_2\chi_{\text{mb}}(\boldsymbol{p}) + w_3\chi_{\text{bi}}(\boldsymbol{p})\,. \qquad (6)$$

As all outlier criteria are based on the k-nearest-neighbor graph, it is computed once and cached during the computation of χ.

Depending on the scanning technique, outliers may occur in small clusters. In this case, χ_{pl} and χ_{mb} tend to fail to detect the clustered outliers correctly. In order to make them suitable for clustered outliers, a maximum cluster size l can be defined by the user. Subsequently, all k-nearest-neighbor queries will discard the first l neighbors, returning the $(l+1)$st to $(l+k)$th neighbor instead. This effectively increases the robustness against clustered outliers while maintaining the basic functionality of the outlier criteria.

5.4. MLS Smoother

Smoothing is an elementary editing operation. It can be used for noise reduction, to smooth-out high-frequency details, such as small artifacts like spikes and ripples, or to soften creases created during the editing process. Various smoothing operators have been proposed, partly with feature-preserving properties.

Given the unpredictable quality of input data, we decided against locally adapting filters, as they still tend to amplify scanning artifacts. We implemented a simple, more robust, filter based on MLS projection, leaving the treatment of features to the user's control by confining the operation to the volumetric brush selection.

(a) (b) (c)

Figure 9: *The MLS smoother tool.* (a) *Fine surface details.* (b) *Smoothing with* $\alpha = 0.8$. (c) *Detail enhancement for* $\alpha = -0.75$.

The *MLS smoother tool* works by shifting point positions towards the corresponding MLS surface. For each point $\boldsymbol{p}$, its MLS projection $\boldsymbol{p}'$ is computed. A user-adjustable blending parameter α defines how far $\boldsymbol{p}$ is to be moved towards its "smoothed" position $\boldsymbol{p}'$. The point is finally set to

$$\boldsymbol{p}_{\text{smoothed}} = (1-\alpha)\boldsymbol{p} + \alpha\boldsymbol{p}'\,. \qquad (7)$$

An associated normal is filtered analogously, blending the original normal with the normal of the MLS surface. Parameterization of the MLS kernel function, as described in Section 4.2, allows the user to adjust the depth-pass characteristic of the MLS projection.

An additional user parameter D allows to attenuate the tool's effect towards the selection border. Within distance D to the border, α is weighted by a blending polynomial to vanish at the border. A point's distance to the border is defined as the distance to its nearest neighbor outside the selection.

α is usually set to values within $[0,1]$, corresponding to strong, or no smoothing, respectively (see Figure 9(b)). Alternatively, following the concept of USM filtering [10], one may set α to negative values, corresponding to a detail (and noise) enhancement (see Figure 9(c)). This is a useful feature, however, for larger absolute values of α, surface self-penetration can occur.

5.5. Point Relaxation

Scanned models may contain regions of uneven point distribution. While some editing operations may change the point distribution directly, raw scan data will be unevenly sampled wherever point samples are missing due to scanning artifacts. Merging of depth-maps also produces an uneven point distribution. However, a uniform distribution of the surfels is often required to guarantee a hole-free rendering of the surface.

To achieve an even distribution of the surfels we employ a

© The Eurographics Association 2004.

particle simulation as described in Section 4.3. The attributes of the relaxed surfels, such as the color, are interpolated from the attributes of the k-nearest original surfel neighbors.

The particle simulation can also be used to close small holes, as the repelling force will distribute the surfels over uncovered areas.

5.6. MLS Spray Can

Complementary to the eraser, the *MLS spray can tool* was introduced in order to fill small holes in the geometry. It randomly creates points inside the brush volume and projects them onto the MLS surface in the brush's vicinity.

A projected point $\boldsymbol{p}$ is added to the surface whenever the surrounding splat coverage is below a certain threshold. The local coverage is estimated by determining the ratio between the average distance $\bar{d}$ of $\boldsymbol{p}$ to its k-neighbors and the mean splat radius $\bar{r}$ of its neighbors. $\boldsymbol{p}$ is added if

$$\frac{\bar{r}}{\bar{d}} < 1\,. \tag{8}$$

Consequently, the MLS spray relies on valid splat radii. When importing a model, we initially compute splat radii using a local surface analysis as proposed in [15], based on a Voronoi diagram of the point cloud.

If a new point is added to the surface, its normal is adopted from the MLS surface. All other surfel attributes, e.g. color and reflectance properties, are determined by interpolating attributes from neighboring surfels.

Application of the spray can tool may result in a roughly uniform point distribution (see Figure 12). Eventually, the point distribution has to be relaxed using the point relaxation tool (see Section 5.5).

5.7. Automatic Hole Filling

While the MLS spray can tool introduced above is very effective for filling small holes, it still remains a tedious process to create a complete watertight model when larger and more complex holes occur in the acquired point cloud. This is frequently the case, however, as line-of-sight constraints, difficult surface reflectance properties, or extensive noise- and outlier removal, can lead to a highly incomplete representation of the model surface (see also Figure 17). As presented in Section 2, many automatic hole filling algorithms exist.

5.7.1. Volumetric Diffusion

We extend the volumetric diffusion method by Davis et al. [6] to point-sampled models by replacing the distance field estimation of [6] by an MLS projection step as proposed in [18]. The distance field is computed on a regular 3D grid that encloses the model surface (see Figure 10). At each grid point we compute the signed distance to the MLS

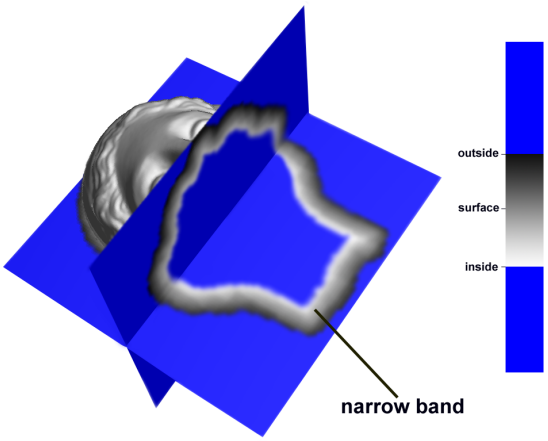


Figure 10: *Volumetric diffusion. Slices of the distance volume reveal the narrow band.*

surface defined by the given input point set. To efficiently represent this volumetric grid we use an octree data structure similar to [8]. This method makes use of binary location codes to address octree cells, allowing for fast point location and efficient neighborhood queries.

We further reduce memory and computation costs by only representing the distance field in a narrow band around the surface, similar to level set methods [14]. We detect holes in the distance field using the classification method of [6]. Distance values on the boundary of holes can then be extrapolated by applying an iterative convolution operator until all holes of a user-specified size are filled. More details on this diffusion process can be found in [6].

To convert the distance field back to an explicit point-sampled representation we either apply a contouring method similar to marching cubes [12], or use a particle simulation as described in Section 4.3. In the latter case, the MLS projection that keeps the particles on the surface is replaced a projection based on gradient decent that moves particles to the zero set of the signed distance field. Normals of the newly generated points can also be directly estimated from the distance field gradient.

The user interface supports a fine-tuning of the algorithm (see Figure 11). Though, using the default parameters the automatic hole filling tool is robust and easy to use.

6. Results

The toolbox has extensively been used by half a dozen people to clean various models acquired with different scanning technologies. We cleaned models acquired with a CyberWare® laser range scanner, a single-shot structured light scanner by 3Q Technologies Ltd., and a phase-shift structured light scanner that allowed us to change various

© The Eurographics Association 2004.

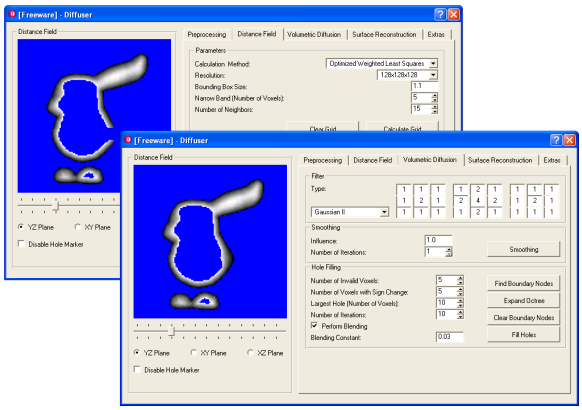
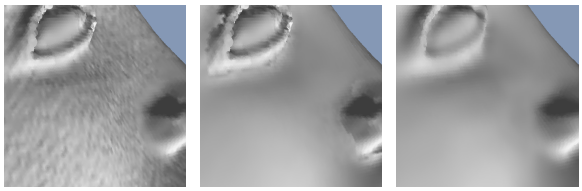


Figure 11: *The user interface of the automatic hole filling tool allows to fine-tune the algorithm. The volumetric representation can be previewed before surface reconstruction.*



(a) (b) (c)

Figure 14: *Selective noise removal using the MLS smoother.* (a) *Noisy input surface.* (b) *Smoothing of a sub-set of surfels, excluding high-frequency details.* (c) *Subsequently, global smoothing of the model.*

scanning parameters. In this section we present some exemplary situations during the model cleaning process.

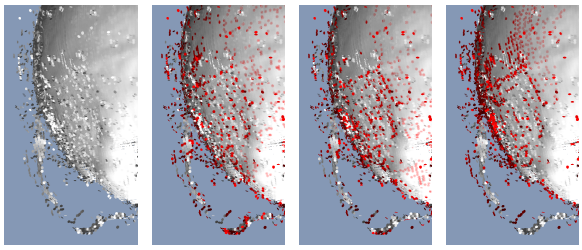
Our general experience is that the simpler, more interactive tools are typically used at the beginning of the cleaning process, whereas the more complex, semi-automatic tools are applied towards the end of the procedure.

It turned out that the simpler tools are often used in combinations to achieve a desired effect. Figure 12 shows how the MLS spray can tool and point relaxation are used to manually fill a hole in a surface.

A similar combination can be used to remove undesired bumps from a surface. Figure 13 shows how the eraser, the MLS spray, point relaxation, and the MLS smoother work together to remove a bump from a surface.

In combination with the selection tool, the MLS smoother can also be used to smooth selected surface parts while preserving details, see Figure 14.

When applying the outlier removal the three different elementary outlier criteria showed to be differently suited depending on the situation (see Figure 15). The plane fit criterion is best suited to detect outliers in a noisy reconstruction of a smooth surface. It produces poor results around small features and creases, as the orientation of the fitted plane becomes instable. The miniball criterion proved to be more robust, even around high-frequency details, but in contrast to the plane fit criterion it shows a poor outlier detection for points that hover close to a smooth surface.



(a) (b) (c) (d)

Figure 15: *Three different outlier classifiers. Potential outliers marked in red.* (a) *Raw scanned geometry.* (b) *Classification using the miniball criterion.* (c) *Plane-fit criterion.* (d) *k-nearest-neighbor graph criterion. All criteria were thresholded to classify 7% of the surfels as outliers.*

In comparison with the previous two, the criterion based on nearest-neighbor reciprocity shows the most robust outlier classification. It is equally sensitive around smooth and detailed regions. However, it consistently yields erroneous outlier classifications around manifold borders (see Figure 16).

Obviously, each criterion is advantageous in different situations. The outlier removal tool allows to confine the outlier detection to certain areas for the model and to weight the criteria according to the local situation.

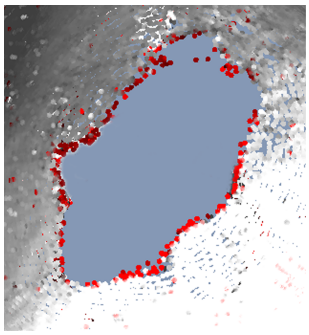
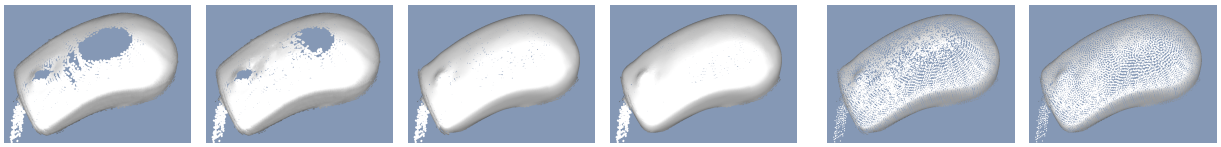


Figure 16: *The nearest-neighbor outlier criterion performs poor around manifold borders.*

In order to test the robustness of the automatic hole filling tool, we used a structured light scanner to scan a furry toy reindeer (see Figure 17). Fur is one of the most difficult materials to be scanned with optical methods. Consequently, the scan shows severe noise and a lot of outliers. Outlier removal leads to a very sparse object reconstruction. However, as shown in Figure 17, the hole filling tool is still capable of producing a water-tight model. Only above the top of the model, the volumetric diffusion had to be constrained in order to get a closed surface.

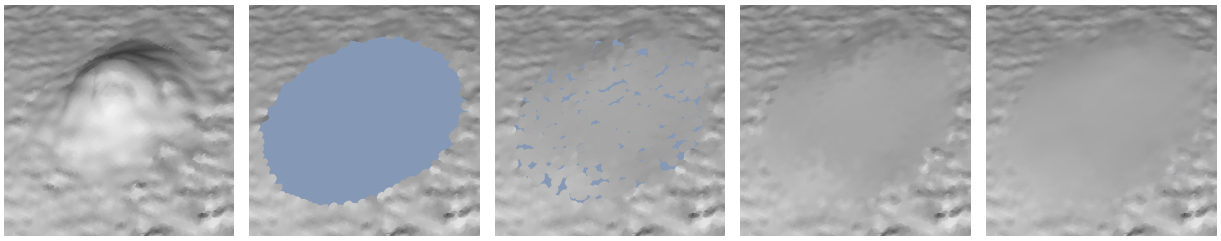
Figure 18 shows an application of the automatic hole fill-

© The Eurographics Association 2004.



(a) (b) (c) (d) (e) (f)

Figure 12: *Manual hole filling using the MLS spray can tool.* (a) *A poor scan of a computer mouse, containing a hole in the surface.* (b,c) *Gradually filling the hole using the MLS spray can.* (d) *Point relaxation improves the point distribution.* (e,f) *Versions of (c) and (d) with reduced splat radii to reveal the point distribution.*



(a) (b) (c) (d) (e)

Figure 13: *Removal of an undesired bump.* (a) *Close-up of the original data.* (b) *The eraser is used to stamp out a hole.* (c) *Using the MLS spray can, the hole is filled.* (d) *Point relaxation redistributes points.* (e) *Locally applying the MLS smoother, attenuating its strength towards the border of the hole. Note the smooth transition of the novel surface to the noisy surrounding.*

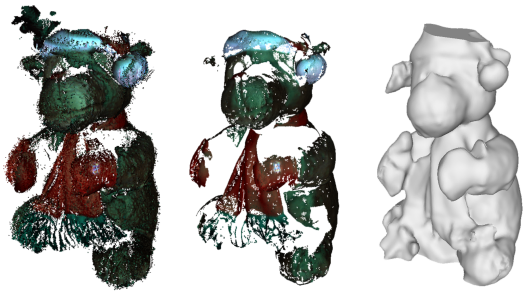
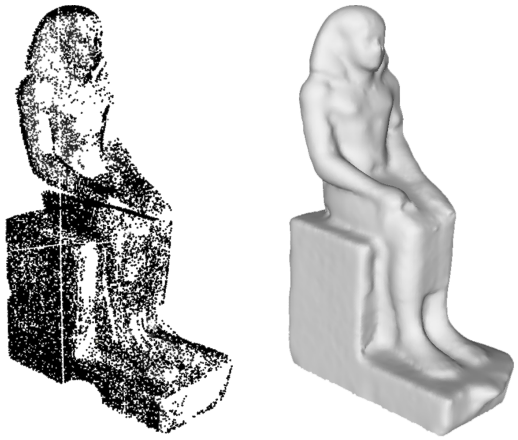


Figure 17: *Robustness of the volumetric diffusion tool.* Left: *The furred object surface produces severe noise and outliers.* Center: *After the outlier removal, only little object points are left.* Right: *The volumetric diffusion tool still reconstructs a water-tight model.*



(a) (b)

Figure 18: *Egyptian sculpture scanned at the British Museum.* (a) *Input scan with varying sampling density.* (b) *Application of the volumetric diffusion tool.*

ing tool to a scan with largely varying sampling density. The model has been scanned by INSIGHT [9] at the British Museum, London.

7. Conclusion

We presented a cleaning toolkit for the post-processing of raw scanner data. It is entirely based on point-based modeling techniques, which are given at hand in the form of simple, interactively controllable, tools. We introduced the underlying techniques and discussed the design principles leading to the presented set of tools.

The tools include an eraser tool, low-pass filters, and various re-sampling and hole-filling tools. We proposed three different outlier criteria that were incorporated in an outlier detection tool. We presented an adaption of the volumetric

© The Eurographics Association 2004.

diffusion algorithm to point-sampled data, using it to build an automatic hole-filling tool.

We evaluated the toolbox, cleaning various objects acquired with different scanner technologies. It proved to be versatile and well-adaptable, as the tools could interactively be re-combined depending on the situation. Most operations are robust against sampling artifacts and do not impose any topological constraints on the data. Future experiences will show whether the toolbox has to be extended. Possible extensions may be additional filter tools, or the integration of texture synthesis into the MLS spray can. As a Pointshop3D plugin the toolbox rounds off a point-based work flow for the processing of scanned 3D surface data.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. Point set surfaces. In *Proceedings of IEEE Visualization*, pages 21–28, San Diego, CA, October 2001.

[2] S. Arya and D. M. Mount. Algorithms for fast vector quantization. In *Data Compression Conference*, pages 381–390. IEEE Computer Society Press, 1993.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, pages 509–517, 1985.

[4] W. Böhler, G. Heinz, A. Marbs, and M. Siebold. 3d scanning software: an introduction. In *CIPA Heritage Documentation, International Workshop on Scanning for Cultural Heritage Recording*, pages 47–51, Corfu, Greece, September 2002.

[5] J. C. Carr, R. K. Beatson, J.B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 67–76, Los Angeles, CA, August 2001.

[6] J. Davis, S. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *First International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 428–861, June 2002.

[7] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, pages 209–226, 1977.

[8] S. Frisken and R. Perry. Simple and efficient traversal methods for quadtrees and octrees. *Journal of Graphics Tools*, 7(3), May 2003.

[9] The institute for study and implementation of graphical heritage techniques (INSIGHT). URL: http://www.insightdigital.org/.

[10] Jain A. K. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[11] D. Levin. Mesh-independent surface interpolation. In *Advances in Computational Mathematics*, 2001.

[12] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics*, SIGGRAPH '87 Proceedings, pages 163–169, San Francisco, CA, October 1987.

[13] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. In *Computer Graphics*, Siggraph 2003 Proceedings, pages 463–470, San Diego, Ca, July 2003.

[14] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton–jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[15] M. Pauly. *Point Primitives for Interactive Modeling and Processing of 3D Geometry*. PhD thesis, Department of Computer Science, ETH Zurich, 2003.

[16] M. Pauly, M. Gross, and L. P. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization '02*, pages 163–170. IEEE Computer Society, 2002.

[17] M. Pauly and M. H. Gross. Spectral processing of point-sampled geometry. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 379–386, Los Angeles, CA, August 2001.

[18] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. In *Computer Graphics*, Siggraph 2003 Proceedings, pages 641–650, San Diego, CA, July 2003.

[19] Pointshop3D. URL: http://www.pointshop3d.com/.

[20] M. Teschner, B. Heidelberger, M. Müller, D. Pomeranerts, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proc. Vision, Modeling, Visualization VMV*, pages 47–54, 2003.

[21] G. Turk. Re-tiling polygonal surfaces. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 55–64. ACM Press, 1992.

[22] J. Verdera, V. Caselles, M. Bertalmío, and G. Sapiro. Inpainting surface holes. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pages II: 903–906, September 2003.

[23] E. Welzl. *Smallest enclosing disks (balls and ellipsoids)*, volume 555 of *Lecture Notes Comput. Sci.* Springer-Verlag, 1991.

[24] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: An interactive system for point-based surface editing. In *Computer Graphics*, SIGGRAPH 2002 Proceedings, pages 322–329, San Antonio, TX, July 2002.

© The Eurographics Association 2004.