Spectral Processing of Point-Sampled Geometry

Mark Pauly
pauly@inf.ethz.ch

Markus Gross
grossm@inf.ethz.ch

ETH Zürich

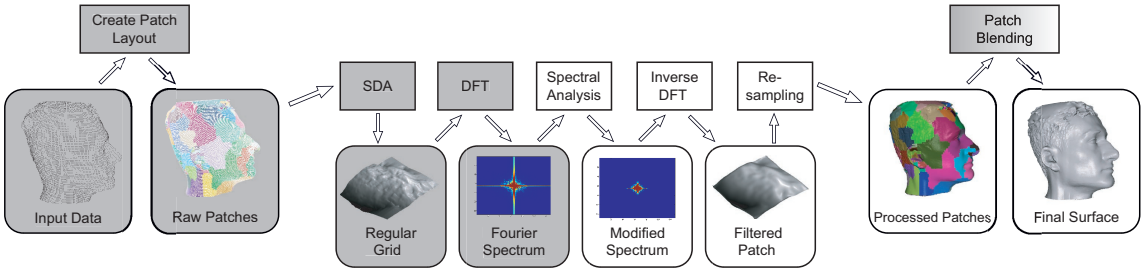


Figure 1: Spectral processing pipeline. Processing stages are depicted as rectangles, rounded boxes represent input/output data of each stage. Gray background color indicates the preprocessing phase.

Abstract

We present a new framework for processing point-sampled objects using spectral methods. By establishing a concept of local frequencies on geometry, we introduce a versatile spectral representation that provides a rich repository of signal processing algorithms. Based on an adaptive tesselation of the model surface into regularly resampled displacement fields, our method computes a set of windowed Fourier transforms creating a spectral decomposition of the model. Direct analysis and manipulation of the spectral coefficients supports effective filtering, resampling, power spectrum analysis and local error control. Our algorithms operate directly on points and normals, requiring no vertex connectivity information. They are computationally efficient, robust and amenable to hardware acceleration. We demonstrate the performance of our framework on a selection of example applications including noise removal, enhancement, restoration and subsampling.

Keywords: Signal processing, spectral filtering, subsampling, Fourier transform, point-based representations

1 Introduction

Today's range sensing devices are capable of producing highly detailed surface models that contain hundreds of millions of sample points. Due to a variety of physical effects and limitations of the model acquisition procedure, raw range datasets are prone to various kinds of noise and distortions, requiring sophisticated processing methods to improve the model quality. In spite of recent advances made in mesh optimization, traditional mesh processing algorithms approach their limits, since triangle primitives implicitly store information about local surface topology including vertex valence or adjacency. This leads to a substantial additional overhead in computation time and memory costs. With increasing model size we thus experience a shift from triangle mesh representations towards purely point-based surface descriptions. For instance, recent work concentrated on point-based rendering pipelines [18, 19], where point samples without connectivity are proposed as rendering primitives. Surprisingly, however, little work has been done so far on direct processing or manipulation of point-sampled geometry. In this paper we present a new framework for spectral analysis and processing of point-sampled objects. The method operates directly on irregular point sets with normals and does not require any a priori connectivity information. Our framework extends so-called windowed Fourier transforms - a concept being well known from signal processing - to geometry.

The Fourier transform is a powerful and widely used tool for data analysis and manipulation. In particular, image processing techniques successfully exploit frequency representations to implement a variety of advanced spectral processing algorithms comprising noise removal, enhancement, feature detection and extraction, up/down-sampling, etc. [7]. Extending this approach to general geometric models is difficult due to a number of intrinsic limitations of the conventional Fourier transform: First, it requires a global parameterization on which the basis functions are defined. Second, most FT algorithms require a regular sampling pattern [17]. These prerequisites are usually not satisfied by common discrete geometry, rendering the standard Fourier transform inoperable. A further limitation of traditional Fourier representations is the lack of spatial localization making it impractical for local data analysis. We will show how these limitations can be overcome and present a generalization of the windowed FT to general 2-manifolds. The basic idea behind our framework is to preprocess the raw irregular point cloud into a model representation that describes the object surface with a set of regularly resampled height fields. These surface patches form "windows" in which we compute a discrete Fourier transform to obtain a set of local frequency spectra. Although being confined to individual surface patches, our windowed FT provides a powerful and versatile mechanism for both local and global processing. The concept of frequency on point-sampled geometry gives us access to the vast space of sophisticated spectral methods resulting from tens of years of research in signal processing.

In this paper we will focus on two classes of such methods: Spectral filtering and resampling. We will point out how sophisticated filtering operations can be implemented elegantly by analyzing and modifying the coefficients of the frequency spectrum. Possible applications include noise removal, analysis of the surface microstructure and enhancement. Further we present a fast algorithm for adaptively resampling point-based geometry, using the spectral representation to determine optimal sampling rates. This method is particularly useful for reducing the complexity of overly dense point-sampled models. By using FFT and other signal processing algorithms our framework is efficient in computation and memory costs, amenable to hardware acceleration and allows us to process hundreds of millions of points on contemporary PCs.

1.1 Previous Work

Extending the concept of frequency onto geometry has gained increasing attention over the last years. Conceptually, this generalization can be accomplished by the eigenfunctions of the Laplacian. Taubin [20] pioneered spectral methods for irregular meshes using a discrete Laplacian to implement iterative Gaussian smoothing for triangle meshes. This method has later been improved by Desbrun et al. [4] who tackled the difficulty of discretizing a geometric Laplacian by introducing curvature flow for noise removal. Kobbelt [13] presented a novel concept for multiresolution variational fairing and modeling, where high mesh frequencies are attenuated by iteratively solving discretized Laplacian equations. While being based on signal processing methodology, these algorithms do not compute an explicit spectral representation of the object surface. Hence typical filters such as Gaussian smoothing have to be implemented in the spatial domain. In contrast, our method generates a set of local Fourier spectra that can be explicitly analyzed and manipulated. This supports more powerful filtering, e.g. least-squares optimal or inverse, feature enhancement and Fourier sampling. Specifically, we can examine the power spectrum of the surface signal to estimate optimal filter parameters or determine the noise level present in the data.

Guskov et al. [9] introduced signal processing methods using subdivision and pyramid algorithms. While they achieve qualitatively remarkable effects such as band-pass filtering and enhancement, their notion of frequency is based on detail vectors between different levels of a mesh hierarchy. Our scheme uses the Fourier transform, which efficiently computes a projection into the space of eigenfunctions of the Laplacian. Within this framework concepts like natural vibration modes or spatial frequency are solidly founded in the theory of differential calculus. This allows us to exploit many results from the extensive work on Fourier theory including Sampling or Parseval's theorems. With the former we obtain a profound means to determine optimal sampling rates, while the latter supports local error control.

Lately, Karni and Gotsman [12] introduced a method for spectral compression of triangle meshes that is based on a fixed partitioning of the mesh into submeshes. Effective compression is achieved by a direct decomposition of these patches into the eigenfunctions of the Laplacian. While their notion of frequency is strictly local, the explicit eigenvector computations are expensive and potentially unstable, posing serious limits for the efficient processing of large patch sizes.

All of the above methods focus on triangle meshes, relying heavily on connectivity information between vertices. In contrast, our method is purely point-based, requiring only vertex positions and associated normals. This allows direct processing of scanned data without the need to construct polygonal meshes, making it particularly suitable for the very large models obtained with modern range scanners [15].

1.2 Algorithm Overview

Fig. 1 gives a high-level overview of our spectral processing pipeline. In the first stage we split the point-sampled model into a number of overlapping patches. A patch is defined as a collection of sample points that represents a connected region of the underlying surface. The tesselation is done in such a way that the surface represented by each patch can be expressed as a displacement field over a planar domain. The so generated patch layout forms the basis of our windowed Fourier transform and the following stages operate locally on individual patches. First the patch surface is resampled on a regular grid using a fast scattered data approximation (SDA). Then we apply a Discrete Fourier Transform (DFT) to obtain the spectral representation of the patch surface. Using appropriate spectral filters we can directly manipulate the Fourier spectrum to achieve a variety of effects such as de-noising or enhancement. A subsequent inverse DFT reconstructs the filtered patch surface in the spatial domain. We can then also utilize the spectral information to adaptively resample the patch surface. At the end of the pipeline is the reconstruction stage, where the processed patches are stitched together to yield the final object surface. This requires careful attention at the patch boundaries, where we create a smooth transition by blending the overlapping parts of adjacent patch surfaces. As indicated in Fig. 1, the processing pipeline can be split into two phases: Patch layout generation, SDA and DFT can be separated into a *preprocessing* step. We also precompute the parameter mapping between adjacent patches and the blending function used in the reconstruction. This leaves spectral analysis, inverse DFT, resampling and reconstruction as the actual processing stages. We will now describe the individual stages of the processing pipeline in more detail, following the order depicted in Fig. 1.

2 Creating the Patch Layout

We assume that the input sample points represent a smooth two-manifold of arbitrary topology and possibly multiple connected components. Further we require the sampling to be dense enough in the sense that adjacent points in 3-space with similar normal orientation belong to the same local neighborhood of the surface [1]. The goal is to describe the object surface with a set of patches that can be represented as scalar height fields. To achieve this we grow patches by accumulating adjacent sample points subject to a *normal cone condition*. This criterion states that the aperture angle of the cone spanned by the normals of a patch's sample points is less than π. Bounding the normal cone width guarantees that no foldovers can occur, i.e. that we can bijectively map the patch surface to a height field representation over a planar domain. In practice we choose $\pi/2$ as maximum normal cone width, as this provides a more uniform parameter mapping and thus makes the following scattered data approximation more robust. We compute the normal cone with an adapted version of Gärtner's miniball algorithm [3]. It determines the smallest enclosing sphere of a set of normal vectors interpreted as points on the unit sphere. The vector through the center of the miniball gives the normal cone center and its radius determines the aperture angle (see Fig. 2).

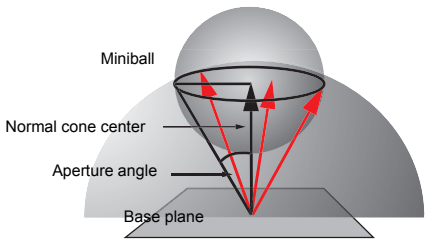


Figure 2: The miniball algorithm provides an accurate estimation of the cone spanned by a set of normal vectors (red).

Our algorithm for generating the patch layout proceeds in two stages: The first stage creates an initial fine-grain patch layout by clustering adjacent sample points, while the second stage merges adjacent clusters into patches using an optimization approach (see also Fig. 4). During this iterative growth we ensure at all times that the normal cone condition is satisfied.

Clustering. We first arrange the sample points in a binary space partition (BSP) tree by recursively splitting the sampling set along the longest axis of its bounding box. The BSP structure implicitly encodes the 3D adjacency information, requiring approx. 10% of the input model size in additional memory overhead. We choose the leaves of the tree, which contain exactly one sample point, as our initial clusters. Now we successively merge clusters with a common parent in the BSP tree, since these are neighbors in 3-space. However, as Fig. 3 illustrates, a cluster has potentially many other neighbors and allowing only sibling clusters to be merged is too restrictive to lead to a useful patch layout. Therefore we stop

the clustering stage as soon as the clusters reach a suitable size (typically 25-100 sample points, depending on model size). We will call the patches created by clustering *leaf patches*, as they are leaves of the final BSP tree.

Patch Merging. At the beginning of the second stage we have to compute local neighborhood information, i.e. for each leaf patch we need to determine a list of all adjacent leaf patches. A leaf patch is confined by six BSP split planes, each of which corresponds to an internal node of the tree. In a first step we collect for each split plane all leaf patches that border on either side of the plane. Then we project the bounding boxes of these leaf patches onto the split plane and check for overlaps of the projections. If an overlap occurs, we mark the leaf patches as neighbors (see Fig. 3).

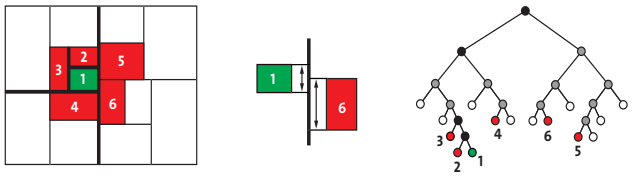


Figure 3: Neighborhood information for leaf patches (2D for illustration). Thick lines (resp. black dots) indicate the BSP split planes that confine the green patch. Patch 1 and 6 are neighbors because their projections onto the split plane overlap. Note that neighbors can be distributed over the whole BSP tree.

Using the adjacency information of the leaf patches, we can now apply a more sophisticated merging technique. The idea is to use an optimization approach that merges patches according to a local quality metric Φ. Let P_i and P_j be neighboring patches such that $P = (P_i, P_j)$ is a potential merge candidate pair. Then $\Phi(P)$ gives a relative measure of the quality of the patch layout obtained after merging P_i and P_j, with small values of Φ indicating a high quality. By iteratively merging the pair with the highest quality gain we can locally optimize the patch layout. Merge candidate pairs are arranged in a priority queue that is ordered by increasing Φ and initialized with all pairs of neighboring leaf patches. Now we successively remove the pair with the highest priority (i.e. lowest Φ) from the queue and merge the two patches if their union satisfies the normal cone condition. Then we update the priorities and neighborhood information of all affected pairs accordingly. Φ is determined using the following formula:

$$\Phi(P) = \Phi_{Size}(P) \cdot \Phi_{NC}(P) \cdot \Phi_{B}(P) \cdot \Phi_{Reg}(P) . \qquad (1)$$

Each factor of Eq. 1 seeks to optimize a specific quality feature of the final patch layout. Since the individual quality measures are difficult to normalize, we combine them in a product to yield Φ.

- Φ_{Size} assigns a high priority to small patches and thus reduces undesirable fragmentation:

  $$\Phi_{Size}(P_i, P_j) = |P_i| \cdot |P_j| ,$$

  where $|P_k|$ is the number of samples in patch P_k.

- Φ_{NC} penalizes the increase in normal cone width of the merged patch $P_i \cup P_j$:

  $$\Phi_{NC}(P_i, P_j) = \alpha(P_i \cup P_j) - max\{\alpha(P_i), \alpha(P_j)\} ,$$

  where $\alpha(P_k)$ is the aperture angle of the normal cone of P_k. This leads to a better adaptation of the patch layout to the local curvature of the underlying surface, since flat regions are quickly merged into large patches, while highly curved regions will be covered by smaller patches.

- Φ_B is introduced to control the boundary of the patches:

  $$\Phi_B(P_i, P_j) = \frac{L(P_i \cup P_j)}{B(P_i) + B(P_j) - B(P_i \cup P_j)} ,$$

  where $L(P_k)$ counts the number of leaf patches of P_k, while $B(P_k)$ counts only those leaf patches that lie on its boundary[1]. Thus Φ_B seeks to minimize the length of the patch boundary relative to the patch area. This will favour roughly circular-shaped patches, which is beneficial for the later SDA and DFT processing stages.

- Φ_{Reg} is used to regularize the patch distribution:

  $$\Phi_{Reg}(P_i, P_j) = \frac{E_s(P_i \cup P_j)}{E_s(P_i) + E_s(P_j)} ,$$

  where $E_s(P_k) = \sum_{l \in N(k)} \sigma \|\boldsymbol{c}_k - \boldsymbol{c}_l\|$ is a spring energy term.

  It is derived by placing a spring with tension σ on each edge from the center $\boldsymbol{c}_k = (c_x, c_y, c_z)$ of P_k to the center of all neighboring patches P_l, $l \in N(k)$.

The merging process terminates as soon as no more patches can be merged without violating the normal cone condition. To have additional control over the granularity of the patch layout, the user can specify a maximum patch size in terms of number of sample points or spatial extent. One could also assign different weights to each of the individual quality measures by using additional exponents in Eq. 1. In practice we found, however, that equal weights generally lead to satisfactory results. Fig. 4 illustrates the two stages of the patch layout generation for the simple example of a sphere. Figs. 13, 15 and 16 show the final patch layout for more complex point-sampled models. Observe how the distribution and shape of the patches adapts to the geometry, i.e. in regions of high curvature we have more and smaller patches than in flat parts of the surface.

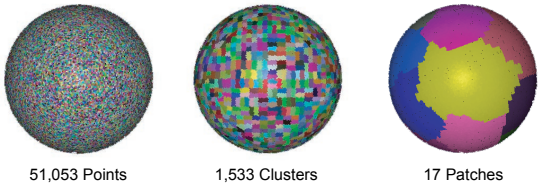


Figure 4: The patch layout is created by first merging sample points into clusters and then merging clusters into patches.

3 Scattered Data Approximation

The patch generation algorithm does not require nor create any connectivity information of individual samples. At this point a patch is simply a set of irregular sample points without any additional knowledge about the spatial relations between them. The goal of the next stage of the processing pipeline is to create a continuous surface representation that describes the patch surface as a scalar displacement field sampled at regular intervals.

Functional Mapping. The first step in doing so is to define the local coordinate frame of the height field representation. We call the plane specified by the center of a patch's normal cone its *base plane* (see Fig. 2). It defines a coordinate transformation T that maps a sample $\boldsymbol{p} = (x, y, z)$ given in world coordinates to $\boldsymbol{p}' = T\boldsymbol{p} = (u, v, h)$, where h is the displacement from the base plane at parameter values (u, v). Then we compute the smallest enclosing box of all (u, v) pairs on the base plane [6], so that we can optimally align the sampling grid to the sample points (Fig. 5).

Overlap. As mentioned before, we need to let patches overlap to handle boundary effects during the reconstruction stage. This is achieved by increasing the size of the parameter rectangle and including all sample points from neighboring patches that map into the enlarged parameter domain (see Fig. 5, right). We check for each boundary point, if it satisfies the normal cone condition. Here

[1] internally we store a patch as a list of leaf patches that constitute the patch, hence L and B can easily be evaluated.

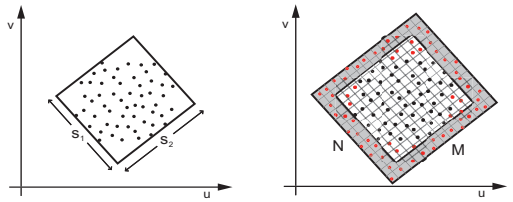


Figure 5: Left: Smallest enclosing box of the interior sample points in the parameter plane. Right: extended parameter domain with regular sampling grid. The red dots indicate boundary points that have been included from neighboring patches.

it has proven useful to increase the maximum normal cone width for boundary points by $\pi/8$, allowing more information from the underlying surface to be included in the overlap region. For the applications of this paper we found an overlap size of 10% of the interior parameter box sufficient for creating a smooth transition at the boundaries during reconstruction.

Regular Sampling. As Fig. 5 illustrates, the sampling pattern on the base plane is in general *irregular*. Standard spectral transforms such as Cosine or Fourier transforms require regularly sampled input data, however. Therefore we apply a fast, hierarchical scattered data approximation, which projects the displacement field onto a regular grid. We use linear B-Spline basis functions centered at each grid point, such that the support of each basis touches the center of its eight neighboring basis functions. Linear B-Splines allow for efficient evaluation due to their compact support, and interpolate the original samples provided the sampling rate is sufficiently high. We utilize the scattered data approximation method presented by Gortler et al. [8] for image based rendering and refer to there for details. As shown in Fig. 6, this algorithm proceeds in three phases:

- *Splatting* computes weighted averages of the sample points to create an initial approximation of the coefficients of the basis functions. Due to the irregularity of the sample points this first approximation may still contain holes, i.e. undefined regions, that need to be filled.
- *Pull* iteratively generates lower resolution approximations through hierarchical convolution filtering.
- *Push* fills the holes in the final patch by successively blending approximations at different resolutions.

We set the grid size $N \cdot M$ proportional to the number n of interior and boundary points of the patch: $N = \lceil s_1 K \rceil$, $M = \lceil s_2 K \rceil$, where $K = \kappa\sqrt{n/(s_1 s_2)}$ with oversampling factor κ (see Fig. 5). For all our models we chose $\kappa = 1$, which leads to an approximation error[1] of less than 0.01%. Note that substantially smaller grid sizes introduce some noticeable low pass-filtering due to the averaging of the splatting phase. As explained above, our patch layout describes a surface by a set of scalar-valued displacement coefficients. A similar approach was taken for displaced subdivision surfaces [14] and normal meshes [10] that achieve staggering mesh compression rates. Both methods, however, require the costly computation of a coarse triangle mesh to obtain the base domain for the displacements. In addition, nontrivial parameterizations are mandatory to keep track of coefficients. This creates a substantial computational overhead, making both representations less suitable for our purposes. The patch layout scheme is much more simple and neither requires triangle meshes nor mesh simplification. The described procedures operate directly on point clouds, making them fast and efficient even for very large datasets (see Table 1).

[1] measured as the RMS error between original sampling points and reconstructed points from the SDA surface.

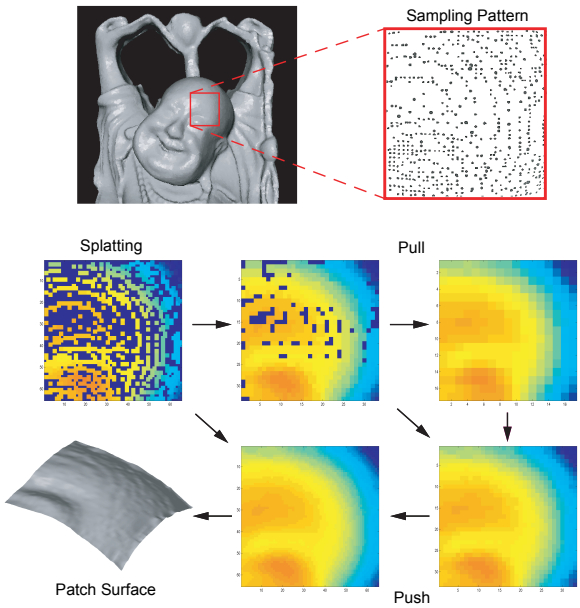


Figure 6: Scattered data approximation for a patch of the Happy Buddha. Different colors illustrate the displacement from the base plane with dark blue pixels indicating holes. The latter are quickly filled during the pull phase by halving the resolution in each dimension. The final patch is created in the push phase, where the arrows indicate which grids are blended.

4 Discrete Fourier Transform

The surface representation created by the SDA describes a point-sampled model with a set of overlapping patches, each of which satisfies the Fourier requirements of regular sampling distribution and Euclidean domain. We can thus apply a discrete Fourier transform (DFT) using a 2D box window function[2] to obtain a spectral decomposition of the surface model. In order to better understand what follows, we give a brief introduction of the DFT, mentioning only those properties that we directly exploit in our algorithms. For more details we refer to textbooks such as [2].

The two-dimensional DFT is essentially a basis transform into the space of eigenfunctions of the Laplacian. Given a real-valued input signal $\mathbf{x}$ defined on a regular grid of size $N \cdot M$, the coefficients of the DFT $\mathbf{X} = F(\mathbf{x})$ can be written as

$$\mathbf{X}_{k,l} = \sum_{n=1}^{N} \sum_{m=1}^{M} \mathbf{x}_{n,m} e^{-j\frac{2\pi}{N}kn - j\frac{2\pi}{M}lm}, \qquad (2)$$

where $j = \sqrt{-1}$ and $(k, l) \in ([1, N], [1, M])$ are the discrete frequencies. Using a 2D Fast Fourier transform (FFT), we can compute the DFT in $O(NM\log(NM))$ operations, instead of $O(N^2M^2)$ operations required for the direct evaluation of Eq. 2 [5]. Fundamental for the implementation of the spectral filters described below is the convolution theorem. It relates a convolution $\mathbf{x} \otimes \mathbf{y}$ of two signals $\mathbf{x}$ and $\mathbf{y}$ in the spatial domain with a multiplication in the spectral domain: $F(\mathbf{x} \otimes \mathbf{y}) = F(\mathbf{x}) \cdot F(\mathbf{y})$. Instead of doing a computationally expensive (filtering) convolution in the spatial domain, we can thus perform a cheap multiplication in the frequency domain using the DFT and its inverse.

Power Spectrum. The power spectrum P is the Fourier transform of the autocorrelation function, i.e. $P(\mathbf{x}) = F(\mathbf{x} \otimes \mathbf{x}^*)$, where the asterisk denotes the complex conjugate. Power spectrum estimation is a widely used tool in data analysis. As illustrated in Fig. 7, it allows to estimate the signal-to-noise ratio and can thus be used to optimize filter characteristics such as cut-off frequency

[2] other windows, e.g. Gaussian or Hanning, can be used too.

or pass- and stop-band. Using the convolution theorem, we can directly compute P from the spectral coefficients, i.e. $P(\mathbf{x}) = |\mathbf{X}|^2$ with $\mathbf{X} = F(\mathbf{x})$.

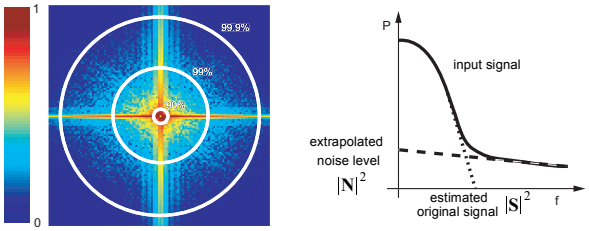


Figure 7: Power spectrum estimation. Normalized logarithmic plot of the power spectrum of a typical patch surface (left). The annotation at each circle indicates the relative amount of power contained within the circle. On the right an idealized illustration for signal-to-noise ratio estimation.

Error Estimation. Another important result from Fourier theory is Parseval's theorem

$$\sum_n \sum_m \mathbf{x}_{n,m}^2 = \frac{1}{NM} \sum_k \sum_l |\mathbf{X}_{k,l}|^2, \tag{3}$$

which relates the signal energy in spatial and frequency domains. We can utilize this property for estimating the error introduced by filtering the spectral coefficients: Suppose $\mathbf{Y} = F(\mathbf{y})$ is a filtered version of the spectrum $\mathbf{X} = F(\mathbf{x})$. Then the L_2 -norm of the difference $\mathbf{x} - \mathbf{y}$ is given by

$$\|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\frac{1}{NM} \sum_k \sum_l |\mathbf{X}_{k,l} - \mathbf{Y}_{k,l}|^2}. \tag{4}$$

Thus we have explicit control over the error introduced when modifying the spectral coefficients.

5 Spectral Analysis

The frequency spectrum obtained by the DFT provides us with a spectral representation of the patch surface. The basis functions in the spectral domain represent natural vibration modes of the surface, thus relating specific surface features to certain frequency intervals. Low frequencies, for instance, represent the overall geometric shape, while high frequencies account for small geometric detail and noise. With these semantics we can perform elaborate filtering operations by manipulating the frequency spectrum. Fig. 8 shows various such filters with the corresponding transfer functions. Low-pass filtering eliminates high frequencies and thus leads to surface smoothing. Observe that the ideal low-pass filter with its sharp cut-off frequency produces the well-known ringing artefacts [7], which are clearly visible as surface ripples in the image. This phenomenon can easily be explained with the convolution theorem: Multiplying the frequency spectrum with the box function of the ideal filter is equivalent to convolving the original surface with a sinc function (see Fig. 9, left image). When using a Gaussian transfer function for surface smoothing, no ringing artefacts occur, since the corresponding filter kernel in the spatial domain is also a Gaussian (Fig. 9, right). The lower left image of Fig. 8 shows a band-stop filter that attenuates middle frequencies. This leads to overall surface smoothing while still retaining the microstructure of the surface material. We can also enhance certain features of the surface by scaling the frequency spectrum appropriately (Fig. 8, lower right).

Signal Restoration. Real imaging systems often introduce some undesirable low-pass filtering since the physical apparatus does not have a perfect delta-function response. This blurring can be reduced with an inverse filter that amplifies high frequencies. Inverse filtering, however, tends to instabilities and is extremely sensitive to noise. To restore the object surface in the presence of blur *and* noise we apply a least-squares optimal filter or *Wiener* filter [11]. Suppose we have a blurred and noisy input signal $\mathbf{x}'$ and

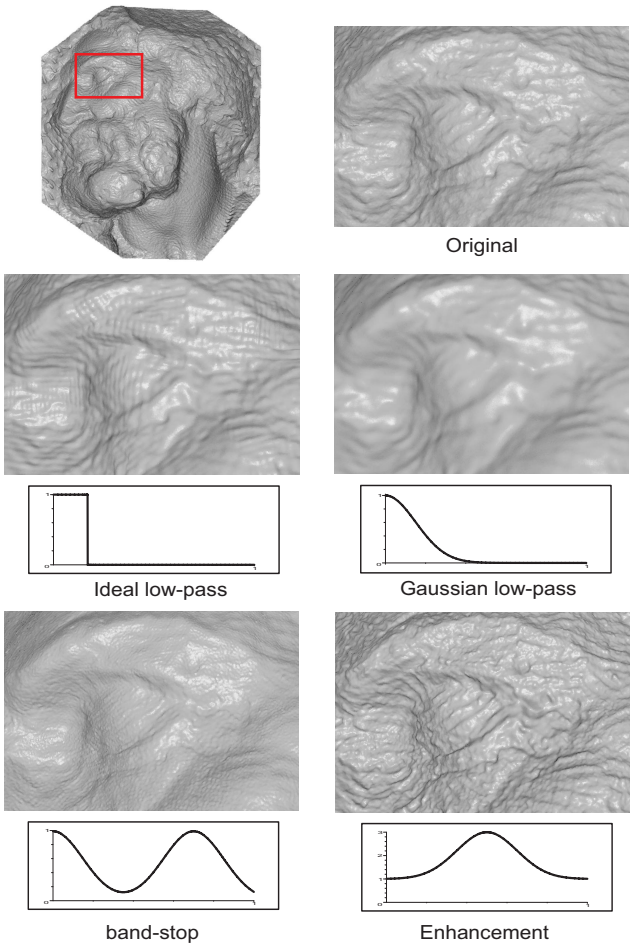


Figure 8: Spectral filters applied to the St. Matthew dataset. The corresponding 2D transfer function is obtained by rotating the shown 1D function around the vertical axis.

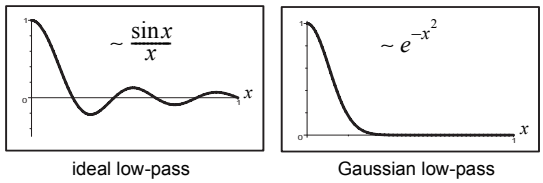


Figure 9: Convolution filter kernels in the spatial domain.

we want to reconstruct the underlying original signal $\mathbf{x}$. Applying the spectral filter function ϕ to the Fourier transform of $\mathbf{x}'$ yields the filtered signal $\mathbf{y} = F^{-1}(\phi \cdot F(\mathbf{x}'))$. The goal is to determine ϕ such that

$$\int |\mathbf{y}(t) - \mathbf{x}(t)|^2 dt = \int |\mathbf{Y}(f) - \mathbf{X}(f)|^2 df \tag{5}$$

is minimized. As shown in [11] this can be achieved by power spectrum analysis, yielding

$$\phi(f) = \frac{|\mathbf{S}(f)|^2}{|\mathbf{S}(f)|^2 + |\mathbf{N}(f)|^2} \cdot \frac{1}{|\mathbf{R}(f)|^2}, \tag{6}$$

where $|\mathbf{S}|^2$, $|\mathbf{N}|^2$ and $|\mathbf{R}|^2$ are the estimated power spectra of the blurred signal, noise and imaging system response, respectively (see Fig. 7, right). Note that effective Wiener filtering relies on an accurate estimation of these quantities, which often requires some knowledge of the system's impulse response (see also Fig. 13).

6 Resampling

After manipulating the frequency spectrum, an inverse DFT takes us back into the spatial domain. If we only want to filter the input model without affecting the sampling pattern and density, we sample the filtered patch surface at the parameter values of the original sample points. However, for many applications it is desirable to have some mechanism for adaptively refining a surface through upsampling or reducing the model size through subsampling. The latter is particularly important when dealing with very large datasets, which often cannot be handled well in their full resolution.

Fourier Sampling. The Fourier spectrum provides us with an elegant way to estimate the optimal sampling rate when subsampling the patch surface. Suppose we have a bandlimited signal $\mathbf{x}$ with Nyquist frequency f_c, i.e. all coefficients associated with frequencies greater than f_c are zero. Then the sampling theorem of Fourier theory states that we can reconstruct $\mathbf{x}$ exactly, if the sampling interval is less than or equal to $1/(2f_c)$. Thus to uniformly subsample the patch surface we proceed as follows: First we low-pass filter the frequency spectrum to obtain a bandlimited signal. Using the power spectrum and error estimation described in Section 4, we adjust the filter parameters to match the desired maximum error. Then we apply the sampling theorem to compute the optimal sampling interval for the filtered signal. Thus we can control the sampling rate by specifying the maximum error tolerance.

Sampling Points and Normals. To determine a patch surface point at arbitrary parameter values, our current implementation uses bilinear interpolation and computes the corresponding normals with first order divided differences. Higher order schemes can easily be implemented as well. Alternatively, we could use the subdivision scheme presented in [14], where the scalar displacements are interpreted as subdivision coefficients.

7 Reconstruction

At this stage of the processing pipeline we need to reassemble the object surface by stitching together the processed patches. Some care needs to be taken here, since individual processing of patches can lead to discontinuities at the patch boundaries. To create a smooth transition between patches we blend the patch surfaces in their regions of overlap. The blending is done by computing a convex combination of corresponding points of neighboring patches using weights given by a precomputed blending function.

Parameter Mapping. To blend points from neighboring patches we need to define a mapping between the different parameter domains in the regions of overlap. Suppose we have an interior point $\boldsymbol{p}_1 = (u_1, v_1, h_1)$ in patch P_1 and that the overlap of patch P_2 also covers $\boldsymbol{p}_1$. The corresponding parameter values (u_2, v_2) can be determined by first mapping $\boldsymbol{p}_1$ to world space using the inverse mapping transform of P_1. This gives us the point $\boldsymbol{q}_1$, which is then projected onto the base plane of P_2. Now we sample P_2 at (u_2, v_2) to obtain $\boldsymbol{p}_2$, which is mapped to world coordinates to yield $\boldsymbol{q}_2$. The blended sample point $\boldsymbol{q}$ is then computed as the convex combination $\boldsymbol{q} = (\omega_1\boldsymbol{q}_1 + \omega_2\boldsymbol{q}_2)/(\omega_1 + \omega_2)$, where ω_1 and ω_2 are the weights given by the blending functions at (u_1, v_1) and (u_2, v_2), respectively. Multiple patch overlaps are handled analogously. To improve performance we can store the parameter mapping in multi-layered texture maps using bilinear interpolation to compute the parameter correspondence of intermediate points.

Blending function. The blending function for a patch is generated by first splatting all interior samples (see Fig. 5) onto a regular grid. This grid is aligned to the sample points in the same manner as the SDA grid, but can be of different resolution. Subsequent convolution filtering with a Gaussian kernel creates a smooth decay to zero at the patch boundary (see Fig. 10). Thus the more we approach the rim of the overlap region of the patch, the smaller will the influence of the sample point be in the convex combination of the blended sample. Splatting can be done using conventional graphics hardware with splat size equivalent to the size of the convolution matrix. The latter is chosen to match the size of the overlap as defined in Section 3, which for all our test cases was sufficient to guarantee hole-free reconstruction. Note that blending

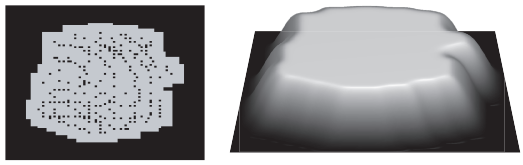


Figure 10: Blending function for the patch of Fig. 6. The left image shows the splatted interior sample points with black dots indicating the sample positions. On the right, the blending function after convolution filtering.

function and parameter correspondence are generated in the preprocessing phase, i.e. operate on the original sample points prior to spectral filtering.

Blending Normals. A smooth boundary transition of normals is achieved analogously to the convex blending used for geometric position. Substantial changes of the shape of the patch surfaces, however, may cause this simple method to fail. Consider the situation of Fig. 11, where the patch processing has created a significant gap between the two surfaces. While the blending of position

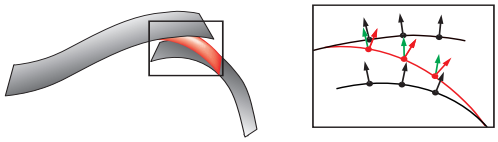


Figure 11: In case of substantial filtering, the blended normals (green) can differ significantly from the correct normals (red).

works fine, the blended normals do not adequately describe the tangent plane of the surface. We detect such cases using a simple conservative heuristic that takes into account the positions of the initial and blended points. The correct normal can then be approximated by sampling a small number of points in the vicinity of the considered sample and fitting a least-squares tangent plane through these points. While computationally more expensive, this normal estimation is rarely required. In all our test cases less than 1% of all normals have been computed in this way, rendering the additional overhead negligible.

Blending the sampling rate. The resampling strategy described in Section 6 uses the sampling theorem to determine the sampling rate for each patch. Since adjacent patches can differ significantly in their spectral representation, this may lead to sharp changes of the sample density at the patch boundaries. For most applications, however, a smooth transition of the sampling rate is preferable. To achieve this, we blend the sampling rate analogously to the blending of geometric positions and normals. This gives us a continuous

Figure 12: Blending the sampling rate at the patch boundary. Left: Continuous sampling rate. Middle: Discretized area weighted sampling rate. Right: Resulting sampling pattern.

function describing the sampling rate, which is then discretized on a regular grid. Each grid value serves as an index into a list of precomputed sampling patterns, generated using Mitchell's algorithm for Poisson disk sampling [16]. Thus we achieve a gradual change of sampling density at patch boundaries (see Fig. 12).

8 Results & Discussion

The filtering and subsampling methods described in Sections 5 and 6 operate locally on individual patches. To achieve global effects we apply the same filter, resp. the same relative error bound for subsampling, to each patch after appropriate scaling of the frequency spectrum.

Fig. 13 shows a Gaussian and a restoration filter applied to a laser range scan of a human head. In our current implementation the parameters of the Wiener filter have to be adjusted interactively by investigating the power spectra of a small number of patches to determine the signal-to-noise ratio. Observe how the Wiener filter preserves geometric features that are smoothed away by the Gaussian.

Interactive local editing operations on the head of the St. Matthew statue are illustrated in Fig. 15. The user can draw a curve on the surface to mark a region of interest (red and blue circles). The patches are split adaptively at this curve and spectral processing is only applied to those patches within the specified area. Note how the patch blending automatically creates a smooth transition between the filtered and non-filtered areas.

Figs. 14 and 16 show subsampling for Michelangelo's David. The original dataset contains 4,128,614 vertices, which have been reduced to 287,165 in the subsampled version, corresponding to approx. 98% of patch signal power. While the sampling of the original model is fairly uniform, the spectral subsampling creates a nonuniform sampling distribution that locally adapts to the geometry. Strictly speaking, the notion of error as established by Eqs. 3 and 4 only holds for the patch interior. The local frequency information in the overlap region - and hence the error - is influenced by the blending function, which in turn results from the convolution process depicted in Fig. 10. Our experimental investigations showed, however, that using the same relative maximum error for each patch leads to a bounded global error and enables intuitive global control. A more thorough analysis of the error behavior at the patch boundary is a main focus of future research.

Model	Head	St.Matthew	David
#vertices	460,800	3,382,866	4,128,614
#patches	256	596	2,966
Computation time (sec.)			
Clustering	1.7	13.5	17.2
Patch Merging	4.8	68.7	61.4
SDA	4.1	32.1	46.3
DFT	0.3	2.9	3.4
Total Preprocess	**10.9**	**117.2**	**128.3**
Spectral Analysis	<0.1	0.2	0.2
Inverse DFT	0.3	2.9	3.4
Reconstruction			
full model	4.6	32.7	57.7
subsampled to 10%	(1.2)	(10.4)	(15.1)
Total	**15.8 (12.4)**	**153 (130.7)**	**189.6 (147)**

Table 1: Timings for the different stages of the processing pipeline (cf. Fig. 1) on a 1.1GHz AMD Athlon with 1.5 GByte main memory.

Performance. Table 1 shows some timing data for our spectral processing pipeline. Note that the bulk of the computation time is spent in the preprocessing stage. Due to its scalar representation, our surface description (comprising SDA and blending grids and parameter mapping) requires less than 40% of the memory of the input model (points and normals) even though no specific compression scheme is applied.

Robustness. An important issue deserving discussion is the effect of a specific patch layout on the final reconstructed surface. Naturally, we want the spectral processing to be invariant under different patchings. While our patching scheme is robust against moderate parameter variations, drastic modifications consequently lead to differences in patch size and shape. Nevertheless, for all examples shown in this paper, we found no perceivable difference when experimentally applying different patch layouts. Of course, if filtering becomes excessive this no longer holds true. If all spectral coefficients are set to zero, for instance, then the patch surfaces will degenerate to the base planes, clearly exhibiting a dependence on the patch layout and the blending function.

Texture and Scalar Attributes. In addition to the geometric information, our pipeline allows to process any attribute data associated with the sample points, such as color or reflectance properties. By including appropriate terms in Eq. 1, these attributes could also be used to control the patch layout scheme.

9 Conclusions & Future Work

We have introduced a spectral processing pipeline that extends standard Fourier techniques to general point-sampled geometry. Our framework supports sophisticated surface filtering and Fourier-based resampling, is very efficient in both memory and computation time and thus allows processing of very large geometric models. Directions for future research include: global error analysis, out-of-core implementation of the processing pipeline, geometry compression, feature detection and extraction, and editing and animation.

Acknowledgements. Our thanks to Marc Levoy and the Digital Michelangelo Project people for providing the datasets of the David and St. Matthew statues. Also many thanks to Simon Rusinkiewicz for making QSplat publicly available.

References

[1] Amenta, N., Bern, M., Kamvysselis, M. A New Voronoi-Based Surface Reconstruction Algorithm. SIGGRAPH 98 Conference Proceedings, 1998.

[2] Bracewell, R.N. The Fourier Transform and Its Applications. McGraw-Hill, New York, 2nd rev. ed., 1986.

[3] Gärtner, B. Fast and Robust Smallest Enclosing Balls. Proc. 7th Annual European Symposium on Algorithms (ESA), Lecture Notes in Computer Science 1643, Springer-Verlag, 1999.

[4] Desbrun, M., Meyer, M. Schröder, P., Barr, A.H. Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. SIGGRAPH 99 Conference Proceedings, 1999.

[5] Dudgeon, D.E., Mersereau, R.M. Multidimensional Digital Signal Processing, Prentice-Hall, 1984.

[6] Freeman, H., Shapira, R. Determining the minimal-area encasing rectangle for an arbitrary closed curve. Communications of ACM, 18, 409413, 1975.

[7] Gonzalez, R.C., Woods, R.E. Digital Image Processing. Addision-Wesley, 1993.

[8] Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F. The Lumigraph. SIGGRAPH 96 Conference Proceedings, 1996.

[9] Guskov, I., Sweldens, W., Schröder, P. Multiresolution Signal Processing for Meshes. SIGGRAPH 99 Conference Proceedings, 1999.

[10] Guskov, I., Vidimce, K., Sweldens, W., Schröder, P. Normal Meshes. SIGGRAPH 00 Conference Proceedings, 2000.

[11] Jain, A.K. Fundamentals of Digital Image Processing, Prentice Hall, 1989.

[12] Karni, Z. Gotsman, C. Spectral Compression of Mesh Geometry. SIGGRAPH 00 Conference Proceedings, 2000.

[13] Kobbelt, L. Discrete Fairing. Proc. of the 7th IMA Conference on the Mathematics of Surfaces '97, 1997.

[14] Lee, A., Moreton, H., Hoppe, H. Displaced Subdivision Surfaces. SIGGRAPH 00 Conference Proceedings, 2000.

[15] Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., Fulk, D. The Digital Michelangelo Project: 3D Scanning of Large Statues. SIGGRAPH 00 Conference Proceedings, 2000.

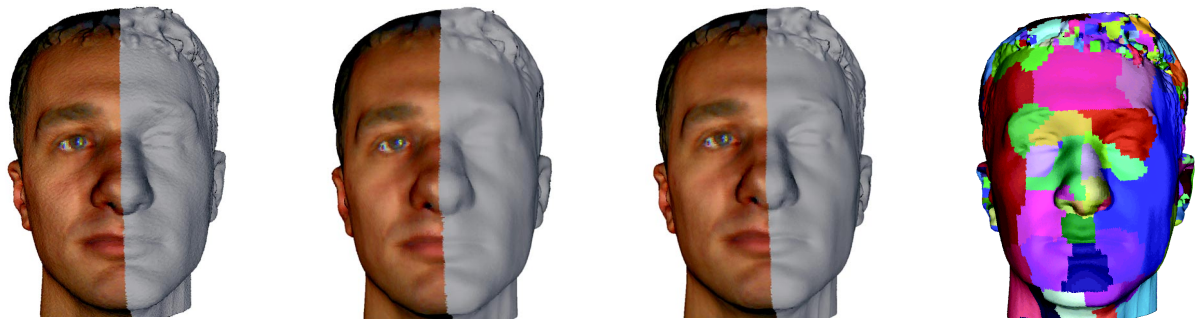
[16] Mitchell, D.P. Generating antialiased images at low sampling densities. SIGGRAPH 87 Conference Proceedings, 1987.

[17] Papoulis, A. Signal Analysis, McGraw Hill, 1977.

[18] Pfister,H., Zwicker, M., van Baar, J., Gross, M. Surfels: Surface Elements as Rendering Primitives. SIGGRAPH 00 Conference Proceedings, 2000.

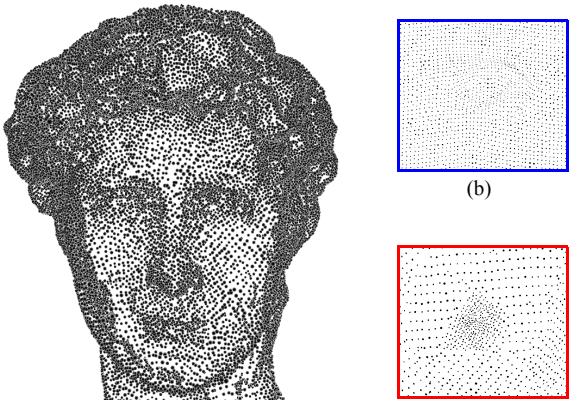
[19] Rusinkiewicz, S., Levoy, M. QSplat: A Multiresolution Point Rendering System for Large Meshes. SIGGRAPH 00 Conference Proceedings, 2000.

[20] Taubin, G. A Signal Processing Approach to Fair Surface Design. SIGGRAPH 95 Conference Proceedings, 1995.



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

Figure 13: Restoration of a blurred and noisy surface model (a), filtered with a Gaussian (b) and a feature-preserving Wiener filter (c). The underlying patch layout is shown in image (d).



(a) (b) (c)

Figure 14: The subsampled head (a) and the zooms of the belly button (b, original) and (c, subsampled) clearly show the nonuniform sampling distributions with more samples concentrated at regions of high curvature.

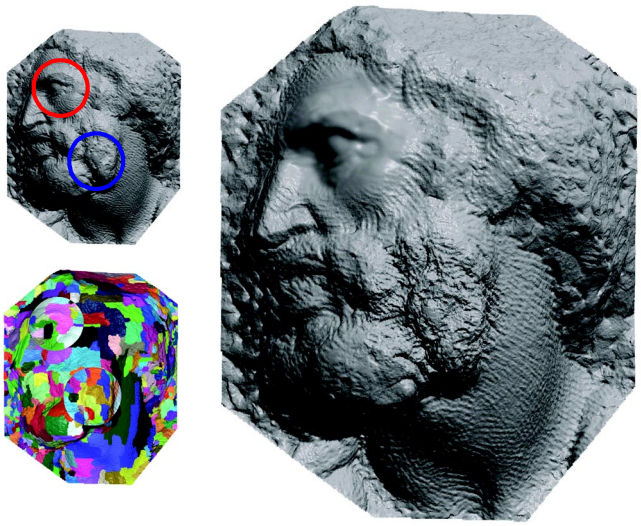
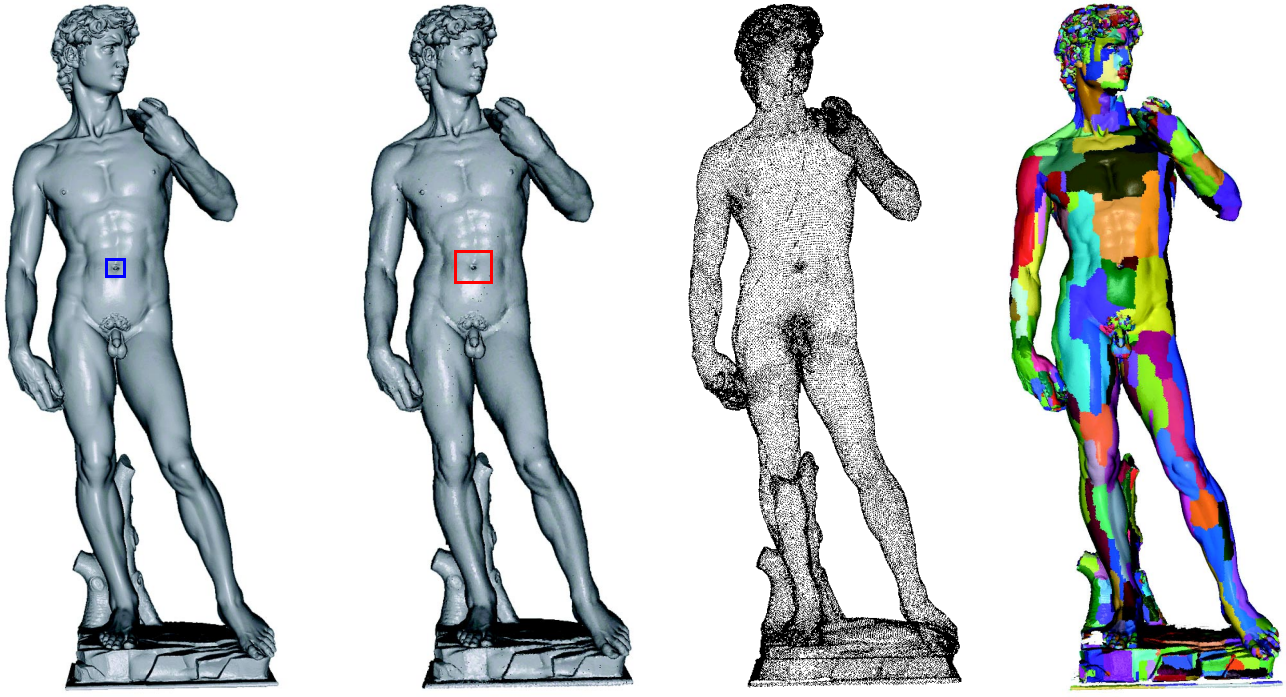


Figure 15: Local smoothing (red circle) and enhancement (blue circle) with adaptive patch layout.



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

Figure 16: Michelangelo's David. QSplat [19] renderings of the original model (a) (4,128,614 vertices) and the subsampled model (b) (287,165 vertices). Image (c) shows the sampling distribution of the latter, while image (d) illustrates the patch layout.