Eurographics Symposium on Rendering - Experimental Ideas & Implementations (2018)
T. Hachisuka and W. Jakob (Editors)

Soft Transparency for Point Cloud Rendering

Patrick Seemann [1], Gianpaolo Palma [2], Matteo Dellepiane [2], Paolo Cignoni [2], Michael Goesele [1]

[1]TU Darmstadt, Germany
[2]Visual Computing Lab - ISTI - CNR, Pisa, Italy

(a) *Surface Splatting* **(b)** *Low quality points removed* **(c)** *One-Layer Transparency* **(d)** *Soft Transparency (Ours)*

Figure 1: *Surface splatting of point data may introduce unwanted or non-realistic occlusion (a). Removal of unwanted points resolves the occlusion, but it also removes data from the scene (b). Rendering the tree with only the first layer transparent results in shallow appearance (c). Our method preserves a notion of thickness with a soft transition from transparent points to regular surface splatting (d).*

Abstract

We propose a novel rendering framework for visualizing point data with complex structures and/or different quality of data. The point cloud can be characterized by setting a per-point scalar field associated to the aspect that differentiates the parts of the dataset (i.e. uncertainty given by local normal variation). Our rendering method uses the scalar field to render points as solid splats or semi-transparent spheres with non-uniform density to produce the final image. To that end, we derive a base model for integrating density in (intersecting) spheres for both the uniform and non-uniform setting and introduce a simple and fast approximation which yields interactive rendering speeds for millions of points. Because our method only relies on the basic OpenGL rasterization pipeline, rendering properties can be adjusted in real-time by user. The method has been tested on several datasets with different characteristics, and user studies show that a clearer understanding of the scene is possible in comparison with point splatting techniques and basic transparency rendering.

CCS Concepts

•***Computing methodologies*** → ***Rendering;*** *Point-based models;* •***Human-centered computing*** → *Visualization techniques;*

1. Introduction

Effective rendering and exploration of raw captured point data, e.g. real-world scenes acquired by a time-of-flight scanner, is still a challenging problem. This is not only due to data size, but also due to other aspects, such as non-uniform sampling and quality of points, or complicated structures to be analyzed. Current point rendering methods tend to treat all data in a point cloud in the same way. Differences in density and quality, as well as complex inner-outer structures, may be represented in an unclear way.

In this paper, we introduce a framework for efficient rendering of mixed-quality point clouds by combining traditional surface splatting with a rendering of semi-transparent spheres of non-uniform density for selected points. The selection of points associated with higher transparency heavily depends on the application. It can be

DOI: 10.2312/sre.20181176

estimated automatically, based on a quality value derived from the normal variation and distance to the nearest neighbors, or assigned manually, based on semantics, for example by setting the external walls of a completely acquired building transparent.

In our rendering method, we visualize selected points using a combination of transparency and "softness": this creates a see-through effect without losing context information of what the points represented originally. Traditional one-layer transparency, i.e., rendering just the foremost surface transparent, has the downside of losing the notion of internal structures and their thickness. Our method keeps all transparent layers and blends them without introducing hard transitions in the opacity or color, thus creating transparency that feels "soft". Furthermore, by using a rasterization approach throughout our pipeline, rendering parameters such as point size, transparency or point segmentation can be easily adjusted by the user without needing to recompute any intermediate representations, as would be the case when using a real-time volume rendering approach which requires computation of acceleration structures in a pre-process.

We motivate our final rendering method by first introducing a "Base-Model" (Section 3.1) for computing color and opacity inside intersecting spheres of non-uniform density. While being physically motivated with regards to light absorption inside volumes with non-uniform density, this exact method is computationally expensive as it relies on the traditional Order-Independent Rendering pipeline using a dynamic A-Buffer and fragment list sorting as proposed by [MCTB12]. Then, we introduce a first approximation with the "Concentric-Model" that substitutes a sphere of non-uniform density with multiple concentric spheres of uniform density. We use a normalization factor to obtain a smooth opacity transition within concentric spheres. Finally, we develop the simplified "Express-Model" as an evolution of the "Concentric-Model" with only one sphere as described in Section 3.3. It relies on pre-sorting the point cloud and uses only one geometry pass to render transparent points which makes it suitable for rendering millions of points interactively.

The method has been tested on a number of point cloud datasets, including laser-scanned environments and multi-view stereo reconstructions, presenting different issues related to the visualization, like presence of vegetation, high variations in data density or hard-to-inspect internal structures. Visual results and a user study (see Section 5.3) show that our contribution provides a more comprehensible rendering.

To summarize, the proposed method is:

- *Simple*: the non-uniform transparency associated to each sphere only depends on the distance from its center, without the need of complex computation.
- *Fast and tunable*: The simplicity of the Express-Model allows to obtain better performance w.r.t. a more accurate underlying density model, without noticeable loss in visual quality. Additionally, the amount of transparency can be changed in real time in order to adapt the visualization to the users' needs.
- *Flexible*: the model visualizes transparency regardless of the type of information it encodes. Hence, it may be used in a number of different applications, from uncertainty visualization to selective rendering.

2. Related Work

Our algorithm is based on the integration of two different visualization techniques: point splatting and transparency.

Point splatting has been thoroughly discussed in [PZVBG00, ZPVBG01, ZRB*04]. Our implementation is based on the three pass splatting as described in [BSK04] and [BHZK05]. The first pass, also called Visibility Pass, renders all points by shifting them by an ε away from the camera to determine which splats will be visible. The Attribute Pass then re-renders the geometry with additive blending, effectively accumulating color and alpha. Lastly, the Normalization Pass normalizes each pixel's color by the accumulated alpha value.

Transparency is a powerful technique used both for visualizing complex scenes and for rendering complex physical effects. To obtain a correct rendering using transparency, a fundamental issue is blending the colors from different layer of geometry in the correct order. To this end, several solutions of Order-Independent Transparency (OIT) were proposed to solve in a correct or in an approximate way the blending of different layers with transparency for real-time or interactive applications. The A-Buffer, introduced by Carpenter [Car84], is a fundamental data structe useful for OIT and has been adapted in different ways ([YHGT10], [VF12]). However, Maule et al. [MCTB12] were the first to propose a method for exact OIT based on constructing a dynamic A-Buffer. It stores for each pixel of the viewport ordered lists of fragments rasterized to that pixel. It is based on three passes: the count of the length of the list for each pixel; the storing of the all the rasterized fragments; the depth sorting of the elements in the list and the blending. We extend this approach for the implementation of two of the models presented in Section 3. In particular, in the first two steps, we store the intersection data of the view ray with the sphere used to approximate the transparent point of the cloud. Then, during the blending, we compute the transparency accumulated along the ray using a physical model of density. The main bottleneck of this approach is the sorting of the per-pixel list, especially when the depth-complexity is large. By modeling explicitly the transmittance along a ray in a volume, Jansen et al. [JB10] removes the sorting of the fragments. In particular, to compute shadowing terms in volume rendering, they traverse the fragment list and estimate the coefficients of a Fourier series that approximates the transmittance along the ray. During blending, they evaluate the approximate function at the depth of the fragment to get its opacity value. Enderton et al. [ESSL11] estimate the transmittance with a stochastic approach without sorting step, using the multi-sample anti-aliasing features on modern GPUs. Salvi et at. [SML11] go one step further selecting a fixed number of fragments most important to the estimation of transmittance. This approach is extended in [MCTB13] by taking the first K fragments and combining the other ones with a simple weighted average. An integration of the point splatting and transparency is presented in [GBP06], where transparency is approximated using a depth peeling approach. Dobrev et al. [DRL10] also use depth peeling, but in an image-space method for rendering points with transparency and shadows. To separate depth layers, they use a per dataset minimum depth parameter, which seems problematic when the point sampling is non-uniform and noisy. For scenes with a high depth complexity a depth peeling approach is

impractical due to the long rendering time to compute all peeling steps. Zhang et al. [ZP06] propose a simple single step algorithm for point splatting and transparent shading. It is based on a precomputed decomposition of the input points into subsets of non-overlapping splats that are rendered independently to different textures and finally blended. While this approach works well for scenes with moderate depth-complexity as presented in their work, it becomes impractical when many transparency layers need to be rendered because one cannot bind enough textures on current hardware. A similar approach is used in [TUH*14], where the point cloud is split in several statistically independent subsets, each rendered using a stochastic approach and then blended together.

Transparency is used in [DWE02] to present a view-dependent model for automatically creating technical illustrations with semi-transparent parts. To do this, a set of rules is extracted from manually created illustrations. Regarding the opacity, one rule states that a points opacity falls off proportionally with the distance to the closest silhouette edge. We use a similar fall-off for rendering spheres with non-uniform opacity guided by the fragments distance to the sphere center. In the context of the integral surface and flow visualization, Hummel et al. [HGH*10] propose angle-based and normal-variation-based methods that differ in the way surface curvature is conveyed through transparency. Both decrease the transparency towards the silhouette of the surface, improving the visualization insight of integral surfaces. Carnecky et al. [CFM*13] present a novel data structure that combines A and G-buffers. This structure allows applying a set of local and nonlocal operators for transparency enhancement improving understanding of complex transparent surfaces.

Similarly, in Direct Volume Rendering (DVR) semi-transparency is used to convey data complexity. Several methods are proposed to improve the perception of internal structures. Volume Illustration [ER00] is a group of techniques that extend standard DVR by enabling the integration of non-photorealistic rendering techniques that modulate opacity and color of each sample based on local features in order to enhance certain structures. Svakhine et al. [SEA09] uses unsharp masking of the depth buffer to achieve an effect visually comparable to ambient occlusion, as it emphasizes cavities through the darkening effect. Bruckner et al. [BG07] integrate halo effects into GPU-based volume rendering to enhance and highlight structures of interest inside the volume, by mapping the halo intensity in color and opacity. Schott et al. [SGM*11] present a technique which integrates a depth of field effect directly into volume rendering, allowing to enhance the area in focus while maintaining a global impression of the whole data set. Volumetric rendering is also used in [SGG15] for visualizing complex particle-based systems using transparency and ambient occlusion. They propose several models to render spheres with uniform and non-uniform density. However, they do not consider the case of multiple intersecting spheres. Even though a Volumetric-Rendering approach can be used to achieve a similar "Soft-Transparency" effect, we implement our method in a way that adapts well to the irregular nature of point clouds. Volumetric-Rendering approaches do not handle well dynamic changes of the rendering properties, such as tuning on the fly the splat radius, the density and any threshold without re-building intermediary volume structures. Our approach is more flexible in this regard and our Express-Model can even be implemented on the web using WebGL since it only relies on standard alpha-blending.

3. Algorithm

Our rendering method takes as input a point cloud with the typical attributes including normal, color and radius as computed by Equation 20. Additionally we precompute a visualization scalar field $\mathcal{F}(\mathbf{p})$ in the range $[0,1]$ defined for each point $\mathbf{p}$ (Equation 21). This scalar field can encode different information which depends on the type of input data. For example the encoded value may be the reliability, confidence and certainty of a 3D scanned point cloud, the semantic category of the objects in the scene, or a simple distance field. The value of this visualization scalar field is used to segment the cloud into two rendering categories according to a user-selected threshold t: the *solid* and the *soft* points. These two clusters of points are rendered in a different way: the solid points have a scalar value below the threshold $\mathcal{F}(\mathbf{p}) \leq t$ and are rendered using the standard three-pass surface splatting. The soft points have a scalar value above the threshold $\mathcal{F}(\mathbf{p}) > t$ and are rendered as spheres with a non-uniform density that determines the final opacity of each fragment of the sphere. The density is highest in the center of the sphere and decreases towards the outside, creating the effect of a "fluffy" ball. To compute the color contribution of each sphere for a given pixel, the viewing ray through that pixel has to be traversed and the density inside each intersected sphere and the relative transparency has to be computed as described in the following sections. Knowing the color and transparency contribution of each sphere along the ray, ordered with respect to the camera viewpoint, the pixels' color can be computed using traditional forward or backward alpha blending. The resulting color of the soft points is then blended with the rendering of the solid points to compose the final image. We use Lambertian shading for all points.

In the next sections we present three different models for the computation of the transparency of each sphere. The first one (Section 3.1) is based on the exact integration of the density inside each sphere while the other two models (Section 3.2 and Section 3.3) propose two approximations which allow us to achieve a visually similar result in a much faster and simpler way. In the rest of the paper we use the following notation:

- C, the RGB-color of the sphere;
- r, the radius of the sphere;
- z^{in} and z^{out}, the depth of the enter and exit intersection points of the view ray with the sphere;
- $z^m = (z^{in} + z^{out})/2$, the middle value along the ray inside the sphere
- d, the distance from the center of sphere to the point z^m which is also the distance from the sphere center to the viewing ray

3.1. Base-Model

For computing the final color of a pixel, all spheres along the viewing ray V have to be traversed and the opacity of each section, as illustrated in Figure 2, has to be computed to perform the correct alpha-blending of the color. As proposed in [Max95] for the direct volume rendering, we compute the transparency $\tau(z_0, z_1)$ when

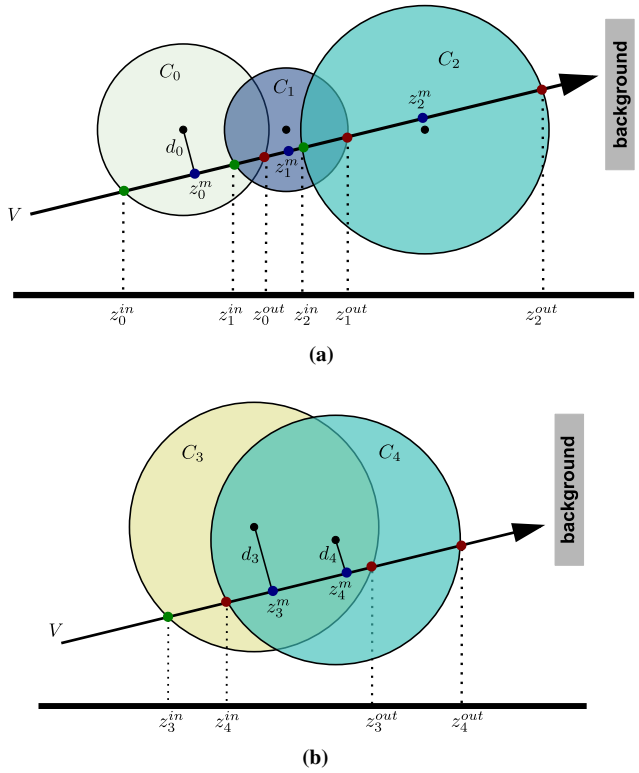


Figure 2: *Examples of density integration along a viewing ray which passes through several intersecting spheres. For each segment along the ray, the density has to be computed to accumulate the opacity and the color using a method presented in Section 3.*

travelling along a ray $r(z)$ inside a sphere from z_0 to z_1 as:

$$\tau(z_0,z_1) = e^{-\lambda\int_{z_0}^{z_1}\rho(z)dz} \tag{1}$$

where $\rho(z)$ is the linear absorption along the viewing ray and λ is a linear factor which can be used for transparency adjustments. $\rho(z)$ can be any density function that is integrable once. Let $P(z)$ denote the integral of $\rho(z)$. The opacity function is then given by:

$$\Phi(z_0,z_1) = 1 - e^{-\lambda(P(z_1)-P(z_0))} \tag{2}$$

For a constant density $\rho(z) = 1$, the opacity function is simply computed as follow:

$$\Phi(z_0,z_1) = 1 - e^{-\lambda(z_1-z_0)}. \tag{3}$$

The idea of the Base-Model is to compute a non-uniform density by integration along the view ray of the reverse distance from the center of the sphere. Considering a single sphere like in Figure 3a, the integral of the density function in the segment $[z^{in}, z^{out}]$ can be split in the sum of two integrals to accumulate the density from the point z^m to the intersection points z^{in} and z^{out} by changing the integral limits:

$$\int_{z^{in}}^{z^{out}} \rho(z)dz = \int_0^{z^m - z^{in}} \rho_{r,d}(z)dz + \int_0^{z^{out}-z^m} \rho_{r,d}(z)dz \tag{4}$$

where the density function for a point $z \in [z^{in}, z^{out}]$ is:

$$\rho_{r,d}(z) = r - \sqrt{z^2+d^2} \tag{5}$$

This formulation of the integral simplifies its computation reducing also the amount of memory needed to store all information. It requires only the depth values along the view ray computed in camera space and not the complete 3D position of the intersection points of the ray with the sphere, as proposed in [SGG15]. The solution of the integral used in Equation 4 is:

$$\int_0^b r - \sqrt{z^2+d^2}dz \tag{6}$$

$$= \left[rz - \frac{z\sqrt{z^2+d^2}}{2} - \frac{d^2\log(\sqrt{z^2+d^2}+z)}{2}\right]_0^b \tag{7}$$

In the case of multiple spheres along the viewing ray, of which some might intersect each other (Figure 2), we compute the density for each segment of the ray delimited by two consecutive ordered hit points with the spheres surface, here z_i^{in} and z_i^{out} with $i \in \{0..2\}$. Depending on the position of the point z^m with respect to the current segment used for the density integration, the integral in Equation 4 can have a different form. In particular if z^m is outside the current segment $[z_0, z_1]$, in the formula we have a single integral. For example in Figure 2b, in the integration along the segment $[z_3^{in}, z_4^{in}]$, since $z_3^m > z_4^{in}$ the accumulated density is:

$$\int_{z_3^m - z_4^{in}}^{z_3^m - z_3^{in}} \rho_{r_3,d_3}(z)dz \tag{8}$$

while in segment $[z_3^{out}, z_4^{out}]$, since $z_4^m < z_3^{out}$ the accumulated density is

$$\int_{z_3^{out}-z_4^m}^{z_4^{out}-z_4^m} \rho_{r_4,d_4}(z)dz \tag{9}$$

In the case of a intersection of multiple spheres, for example the segments $[z_1^{in}, z_0^{out}]$ and $[z_2^{in}, z_1^{out}]$ in Figure 2a, we add the density of each intersecting sphere. For segment $[z_1^{in}, z_0^{out}]$ this would therefore yield the following integral:

$$\int_{z_1^{in}}^{z_0^{out}} \rho(z)dz = \int_{z_1^{in}-z_0^m}^{z_0^{out}-z_0^m} \rho_{r_0,d_0}(z)dz + \int_{z_1^m - z_0^{in}}^{z_1^m - z_1^{in}} \rho_{r_1,d_1}(z)dz. \tag{10}$$

The color C is computed as the average color of the M intersecting spheres weighted by the reverse distance from the center of the corresponding sphere to guarantee a smooth color transition in the intersection region:

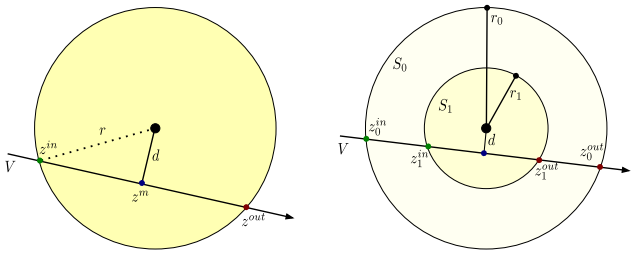
$$C = \frac{\sum_{i=1}^M (1 - d_i/r_i)C_i}{\sum_{i=1}^M (1 - d_i/r_i)} \tag{11}$$

Following the example in Figure 2a, the color of segment $[z_1^{in}, z_0^{out}]$ is equal to:

$$C_{01} = \frac{C_0(1-d_0/r_0) + C_1(1-d_1/r_1)}{(1-d_0/r_0) + (1-d_1/r_1)} \tag{12}$$

3.2. Concentric-Model

Instead of computing the exact integral as described in Section 3.1, the density can be approximated by modeling each sphere as mul-



(a) *Base-Model* **(b)** *Concentric-Model*

Figure 3: *Scheme of the information used by the Base-Model and the Concentric-Model to compute the density accumulate along the ray V.*

(a) *N=25* **(b)** *N=3* **(c)** *N=3 normalization*

Figure 4: *Example of rendering using the Concentric-Model with different number N of spheres (25 and 3) and without (a and b) and with the normalization (c) proposed in Equation 15*

tiple concentric spheres of constant density. This simplifies the integration, as for the uniform case, the density only depends on the distance traveled and can be computed using Equation 3. Suppose a single sphere of radius r is now represented by N concentric spheres S_i with radii $r_i = r - i\frac{r}{N}$, as shown in Figure 3b. Here, the number N is adaptive to the screen space size of the sphere radius, in order to increase the number when the sphere is closer to the camera and needs more details. This means that during interactive navigation the number of concentric elements inside a sphere could quickly change. To avoid temporal inconsistency and artifacts in the rendering, we must force the same accumulate density for every single sphere independently from the number of concentric elements. For this purpose we define a normalization factor μ_i for i-th concentric sphere as follows:

$$\mu_i = \frac{r_0}{N r_i} \tag{13}$$

and slightly adapt Equation 3 for computing the opacity:

$$\Phi(z_0, z_1)_{\text{conc}} = 1 - e^{-\lambda\mu(z_1 - z_0)} \tag{14}$$

For $N \to \infty$, we theoretically have a close approximation of a sphere with non-uniform density. In practice, choosing N such that the radii of the concentric spheres in screen space differ by one pixel is sufficient to create images without visible ringing artifacts. Depending on the sphere size on screen, this may lead to a large N and thus many concentric spheres which results in a performance problem (more in Section 4). It would be favorable to use just a "few" concentric spheres while still making use of the simpler density computation. However, drastically lowering the maximum number N results in sharp color edges near the border of each concentric sphere because the jump in density is too large (Figure 4b). This effect can be reduced by weighting the density by the distance from the center. The normalization factor μ_i is therefore altered in the following way:

$$\mu_i' = \mu_i \left(1 - \frac{d}{r_i}\right) \tag{15}$$

producing a similar effect to the rendering with an higher N (Figure 4c). In this model, in the case of an intersection of multiple spheres, the method sums the density of the spheres and computes the color in the intersection using Equation 11.

For example in Figure 3b the opacity of each segment along the ray V is:

$$\Phi(z_0^{in}, z_1^{in}) = 1 - e^{-\lambda\frac{1}{2}(1-\frac{d}{r_0})(z_1^{in} - z_0^{in})} \tag{16}$$

$$\Phi(z_1^{in}, z_1^{out}) = 1 - e^{-\lambda\frac{r_0}{2r_1}(1-\frac{d}{r_1})(z_1^{out} - z_1^{in})} \tag{17}$$

$$\Phi(z_1^{out}, z_0^{out}) = 1 - e^{-\lambda\frac{1}{2}(1-\frac{d}{r_0})(z_0^{out} - z_1^{out})} \tag{18}$$

3.3. Express-Model

Using our Concentric-Model with the normalization of Equation 15 enables us to render visually similar images compared to our Base-Model. Reducing N to one and setting the opacity of each fragment directly using the distance d from the sphere center, yields our final approximation. Instead of computing the intersections of the view ray with the sphere we fall back to just rendering view-oriented splats that are considered as a disk. In this case, we do not have to manage the density and the color in the intersection between more splats. We experimented with different direct mapping functions of the value d to the opacity of the fragment and we found the following cosine function as the best in regards to the visual similarity to our Base-Model:

$$\Phi_{\cos}(d) = \lambda\frac{(\cos(d\pi/r) + 1)}{2} \tag{19}$$

4. Implementation

Given the input point cloud, the proposed algorithm renders it with two different techniques: standard three-pass surface splatting [BSK04] for solid points and a rendering of spheres using the non-uniform density models presented in Section 3 for soft points. For the Base- and Concentric-Model we use several render passes per frame like the method proposed in [MCTB12] for memory-efficient order-independent transparency, where for each pixel a list of fragments has to be constructed, sorted and blended. During this process, we discard any soft points that are behind fragments of solid points using standard depth buffer testing.

We compute the radius of each point based on the distance to its K nearest neighbors:

$$r_i = 2\sqrt{\frac{\max_{\mathbf{p_j}\in N_i}|\mathbf{p_i}-\mathbf{p_j}|}{|N_i|}} \tag{20}$$

where $\mathbf{p_i}$ is the i-th point and N_i is the set of its K nearest neighbors. The uncertainty scalar field used to separate the point cloud into solid and soft points, e.g., as applied to the RANDERSACKER dataset, is computed based on the geometric uncertainty of each point:

$$\mathcal{F}(\mathbf{p_i}) = r_i \frac{\sum_{\mathbf{p_j}\in N_i}(2-\mathbf{n_i}\mathbf{n_j})^2}{|N_i|}. \tag{21}$$

where $\mathbf{n_i}$ and $\mathbf{n_j}$ is the normal of the point and current neighbor, respectively. The term $2-\mathbf{n_i}\mathbf{n_j}$ maps the scalar product of the normals into the range $[1,3]$ where a value of 1 means the normals point into the same direction. In this case, the points' uncertainty only depends on the sampling density in the neighborhood.

In the first pass, "Fragment Counting", we compute the depth-complexity, that is the number of fragments related to a soft point that are projected over each pixel. This is done by rendering the points with their corresponding radius and increasing a per-pixel atomic counter by the number of intersection samples that will be generated in the next step. For the Base-Model, each point generates two samples (each viewing ray intersects a sphere twice) while for the Concentric-Model this number is $2N$, with N equal to the number of concentric spheres that the point can contain in function of its screen-space radius and the minimum distance among two concentric spheres, as explained in Section 3.2. Knowing the total number of fragments, we create a fragment-buffer, a 4-channel floating-point texture to store the sample lists for all pixels. Since the depth-complexity is view-port dependent, we resize it accordingly for every frame. By computing the prefix sum of the depth-complexity texture, we further get for each pixel an offset into the fragments-buffer that indicates the starting position where to store the array of samples in the next pass. The prefix sum is computed as described in [HSO07] using OpenGL's compute shaders.

The next step, "Fragment Storing", renders soft points and stores the generated samples into the fragments-buffer. We use an additional viewport-sized texture with the index of the next free location in the fragment array of the pixel where we store a new incoming fragment. These indexes are updated atomically at every writing in the per-pixel array. The fragment data is computed by intersecting the pixels' view ray with the sphere represented by the 3D position and the radius of a soft point. The data of each sample is stored in the fragment-buffer using a vector of four 32-bit floating-point numbers. For the Base-Model we store the color (24-bit RGB) of the sample and the distance from the sphere center d (quantized in 8-bit) packed into one float (named *RGBd*), the depth of the intersection, z^{in} or z^{out}, the radius r of the sphere and the midpoint z^m. For the Concentric-Model the data consists of *RGBd*, z^{in} or z^{out} and the normalization factor μ'. We distinguish the enter point from the exit point of the ray by storing the depth as a negative value $-z^{out}$.

The last pass sorts the sample list for each pixel according to their depth and then blends the samples. If the list is smaller than 256 elements, we use merge-sort with two local arrays in the shader. For longer lists, we fall back to an in-place merge-sort which directly operates on the fragments-buffer. The final blending is obtained by the classical forward alpha-blending equation to the sorted list.

In particular, we need to compute for each segment in which the view ray is split by the spheres its opacity and color contribution (for example the segments $[z_0^{in}, z_1^{in}]$, $[z_1^{in}, z_0^{out}]$, $[z_0^{out}, z_2^{in}]$, $[z_2^{in}, z_1^{out}]$, $[z_1^{out}, z_2^{out}]$ in Figure 2). For computing the opacity of a segment according to the Base-Model where multiple spheres intersect, we need to add the density of all spheres that contain the segment. For this reason, we allocate a local array inside the shader to store all open spheres from which the view ray has not yet exited. When entering a sphere, we add it to the array and when leaving it, it is removed. These operations make the Base-Model costly and degrade performance. Furthermore, we need to know the maximum number of intersecting spheres prior to compiling the shader in order to allocate a large enough array. For the Base-Model, sorting and blending have a time complexity of $\mathcal{O}(n\log n + nk)$ with n being the number of samples in the list and k the average number of intersecting spheres.

The Concentric-Model, on the other hand, assumes a constant density and therefore no local array is needed during blending. After the sorting step, the sample list is simply traversed while keeping a running average of the current color and a density-sum. Because each fragment carries information about being an enter or exit point, the average color and density-sum can be updated accordingly while going through the list. However, the Concentric-Model requires more space to store the information of the concentric spheres. The time complexity of sorting and blending is $\mathcal{O}(nm\log(nm) + nm)$ where m is the number of average concentric spheres in the scene.

For both Base- and Concentric-Model, the fragment sorting remains the main bottleneck in the end, especially for scenes with large (i.e. >256) depth complexity.

On the contrary, the Express-Model resolves the performance issues of the other two models by performing the ordering of points only once as a pre-process. Soft points are rendered in a single geometry pass with OpenGL's alpha-blending and depth test enabled (writing is disabled) using the depth map from the previous surface splatting of the solid points. Since we are rendering view-oriented splats (disks) we can pre-sort the points with regards to the six directions of the Cartesian axes and chose the correct sorting for each frame using the axis direction closest to the view direction. This removes the need for the fragment counting, storing and sorting, effectively reducing the time complexity to $\mathcal{O}(n)$ while considerably simplifying the implementation as well.

In the implementation of all three models, the user can change parameters and see in real-time the effects on the rendering. The most important ones are:

- the parameter λ, defined in Equation 1, that can be used to adjust the maximum level of transparency;
- the threshold t to segment the point cloud between solid and transparent regions;

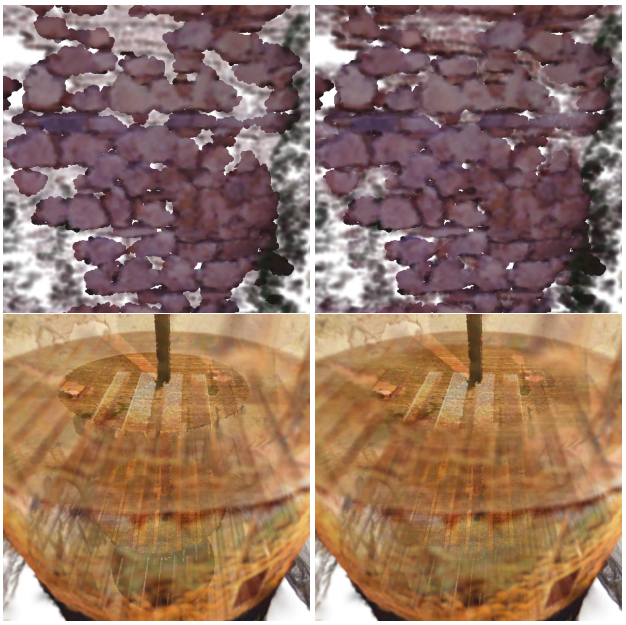


Figure 5: *By modulating the transparency with Equation 22, spheres are rendered with higher density when close to the threshold removing the sharp transition between solid and transparent elements. In both the case the modulation threshold $l_m = 0.2$. (Left) Without transparency modulation. (Right) With transparency modulation. (Top)* CASTLE. *(Bottom)* TOWER.

- the parameter l_m to set in which range of the scalar field $\mathcal{F}$ the algorithm modulates the transparency in order to avoid sharp color edges between a solid region and a close transparent area. This modulation is described in more details in Section 4.1.

4.1. Transparency Modulation

Even if the input scalar field $\mathcal{F}$ of the cloud can have a continuous range in the interval $[0,1]$, the algorithm uses it only for the binary segmentation of the cloud between the solid and the transparent points. Since the amount of transparency of a sphere depends on only its own data, it is possible that in the final rendering we can see a very sharp boundary between the solid and the transparent regions (Figure 5). To solve this problem, the continuous values of $\mathcal{F}$ can be used to modulate the transparency of each segment along the view ray. In particular, given a second threshold $l_m > t$, the new transparency $\Phi_f(x)$ is estimated as a linear interpolation between the actual transparency Φ computed by the model and the value $\lambda(1-d)$, using the scalar field value of the point $\mathcal{F}$ for the computation of the interpolation weight:

$$\begin{cases} \lambda(1-d)(1-W(\mathcal{F}(x))) + \Phi(x)W(\mathcal{F}(x)) & \text{if } t < \mathcal{F}(x) < l_m \\ \Phi(x) & \text{if } \mathcal{F}(x) > l_m \end{cases} \tag{22}$$

where $W(\mathcal{F}(x)) = \frac{\mathcal{F}(x)-t}{l_m-t}$.

Figure 5 shows the results with and without modulation.

5. Results

In order to test the performance of the proposed method, we selected a number of point cloud datasets that provide an overview of features where soft transparency helps in visualization and scene understanding (Section 5.1). Section 5.2 discusses the comparison and performance analysis of the three different models presented in Section 3. Finally, Section 5.3 shows the result obtained with a subjective user-study that evaluates the perceived effectiveness of the method in the understanding of the elements in the scene and their arrangement.

5.1. Datasets

The datasets (shown in Figure 11 and 12) can be divided in two broad groups. The first one includes portions of the point cloud where part of the data is not representing a clear surface, and points in the same volume are characterized by strong variance in normal value or abrupt change of resolution.
The first group is represented seven examples, acquired with terrestrial laser scanner or with photos:

- POMPEII-CAECILIO: this dataset represents a portion of an Insula in Pompeii. The point cloud is characterized by the presence of vegetation inside the courtyard of the house.
- SONGO MNARA: the point cloud, depicting a small portion of a heritage site in Tanzania, presents a big tree covering some building remains.
- RANDERSACKER: this 3D scanning dataset depicts a church surrounded by vegetation
- LAND BUILDING: in this acquisition of a city portion, not only vegetation is present, but also several structures with small features only partially acquired.
- KING MOUNTAIN CAIRN: this scene shows a small monument immersed in a wood, with dense vegetation and grass.
- VINEYARDS: the point cloud represents a portion of a vineyards reconstructed with Multi-View Stereo methods.
- CASTLE: a portion of a castle in Spain acquired by a mixture of 3D Scanning and MVS techniques. The dataset presents vegetation covering part of the external walls of the castle.

For RANDERSACKER and SONGO MNARA we used a per-vertex grayscale color encoding the volumetric obscurance of the vertex estimated with [SPCS16]. The other datasets carry a per-vertex RGB color obtained directly from the acquisition device. Transparency is related to the geometric *uncertainty* of data in a K-neighborhood as described by Equation 21. For POMPEII - CAECILIO, SONGO MNARA, KING MOUNTAIN CAIRN, CASTLE and VINEYARDS we used $K = 24$ while for the LAND BUILDING and RANDERSACKER we used $K = 64$.
The second group of datasets presents more complex structures, where both internal and external elements of a building have been acquired. In these cases, exploration of the datasets is difficult due to occlusions and tight spaces, and it is not trivial to get a good view of the overall structure of the building:

- MATTERPORT: this dataset represents an entire building (external+internal) acquired with RGBd cameras [CDF*17]. This complex environment is difficult to visualize since the external structure hides the complex internal arrangement. The transparency is associated to the structural elements of the house, like

walls, windows, doors, roof, ceiling and pillars. We excluded floors to have a better understating of the different levels of the building. For computing the scalar field $\mathcal{F}(x)$, we used the semantic segmentation available in the original dataset.

- TOWER: the tower is a sub part from the CASTLE dataset. It was acquired from a mixture of 3D scanning and MVS techniques. The peculiar cylindrical shape gave the possibility to link transparency to the distance to the axis, so that internal structure can be easily revealed without fully removing the rest of the structure.

5.2. Comparison and Performance Analysis

All tests were performed on a Windows 10 PC equipped with an i7-3930K CPU and a Nvidia GTX 1080. Figure 6 shows a rendering of three scenes: two synthetic scenes, one with a single sphere and one with two intersecting spheres, as well as the VINEYARDS dataset. The scenes are rendered with a classical point splatting method and with our Soft Transparency algorithm using the models presented in Section 3. For the Concentric-Model we use an adaptive number of spheres by adding a new one every 4 pixels of the screen space radius of the original sphere. The images with the three transparency models are very similar making difficult to distinguish them. This is also confirmed by the difference maps from the Base-Model to the Concentric- and Express-Model showed in Figure 7.

On the contrary, the performance analysis of the three models shows significant differences. Figure 8 shows the trend of the frames-per-second and total number of generated fragments plotted for a free navigation of 50 seconds inside the VINEYARDS dataset. The Base- and Concentric-Model have a higher memory occupancy. The reason is the additional space needed to store the per-pixel lists with the ray-sphere intersection points. They require a variable sized fragment buffer to store the lists and two additional textures for storing per pixel global and relative offsets. The size of the fragment buffer changes during the navigation in function of the depth complexity due to the number of transparent points in the current view. The Express-Model needs only the additional space to store the six index arrays where the points are sorted along the six directions of the three Cartesian axes, that is lower and constant (six times the number of points in the clouds). The differences are large also for the frames-per-second where the Express model is at least $10x$ faster with respect to the other two models. The main bottlenecks for the Base- and Concentric-Model are collecting and sorting the per-pixel fragment lists. With respect to the classical point splatting, the performance penalty of the Express-Models is very small, making the technique suitable for rendering large point clouds interactively, as reported in Table 1.

Dataset	Points	SP	OLT	ST
RANDERSACKER	27.2 M	5.3	2.9	3.6
LAND BUILDING	16.0 M	9.1	5.4	6.6
CAECILIO	10.8 M	14.8	8.0	9.6
MATTERPORT	7.5 M	19.1	12.3	15.5
CASTLE	7.4 M	20.1	11.4	14.5
SONGO MNARA	6.4 M	20.1	12.5	15.7
TOWER	5.5 M	23.8	14.7	18.2
CAIRN	3.9 M	46.4	26.7	31.9
VINEYARDS	1.2 M	196.8	150.1	154.4

Table 1: *Overview of datasets and their rendering-time in frames per second for the Splatting (SP), One-Layer-Transparency (OLT), and our proposed Soft-Transparency (ST) (using the Express-Model). The run-time corresponds to the viewpoints shown in Figure 11 and Figure 12.*

5.3. User Study

The proposed method aims at enhancing the understanding of a scene, hence we setup a user study to compare our solution to current possibilities.

The user study was organized as follows: each user was asked to watch five groups of video sequences "*to evaluate how well they allow you to understand the elements of the scene and their arrangement.*" Showing three different rendering techniques, each video group presented a navigation inside one of the following five datasets: RANDERSACKER, MATTERPORT, CAECILIO, TOWER and SONGO MNARA. For each dataset, we show the video sequences simultaneously on the screen. The point clouds are rendered with the classical point splatting (Point Splatting); rendering the first layer of uncertain points transparently and the remaining ones opaque (One-Layer Transparency) and our proposed algorithm (Soft Transparency) using the Express-Model. The user received no previous information about the rendering methods, and he had the possibility to freely stop the video, watch it again or seek to a specific position.

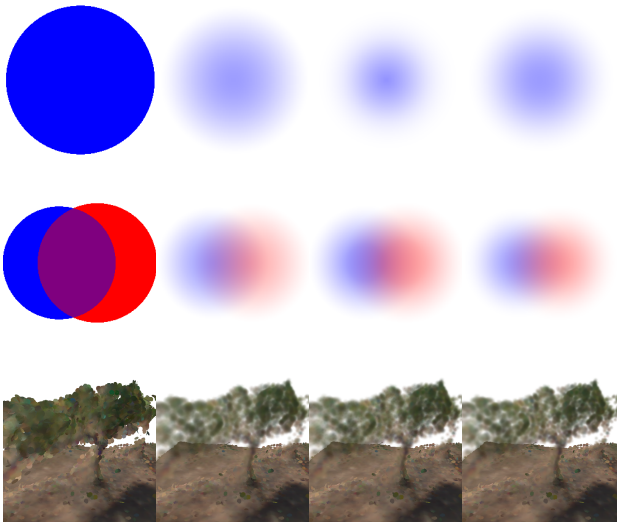
At the end of each video sequence, the user was asked to answer this question: "*Please rate each video according to how well it allows you to understand the elements of the scene and their arrangement.*", by assigning a rating (from 1=worst to 5=best) to each of the versions of the video.

A total number of 66 users, including people with and without expertise in computer graphics, took part in the test. The results for each of the scenes are shown in Figure 10 with the average score and the 95% confidence interval. Soft Transparency proved to be the best method in all the cases. Regarding the other two methods, there's no clear winner between the two solutions: One-Layer Transparency seems more useful when internal structures have to be visualized, while in the case of vegetation or thin structures, everything seems to depend on the disposition of the elements in the scene.

The aggregate chart in Figure 9 shows that the average rating for our method was above 4, with a clear improvement with respect to the other methods. Using the Kolmogorov-Smirnov test, we also reject the null hypotheses that the score values for the three tested methods are from the same distribution (p-value $= 0.0362$ between Point Splatting and One-Layer Transparency, p-value $= 2.14 \times 10^{-21}$ between Point Splatting and Soft Transparency and p-value $= 1.50 \times 10^{-19}$ between One-Layer Transparency and Soft Transparency).

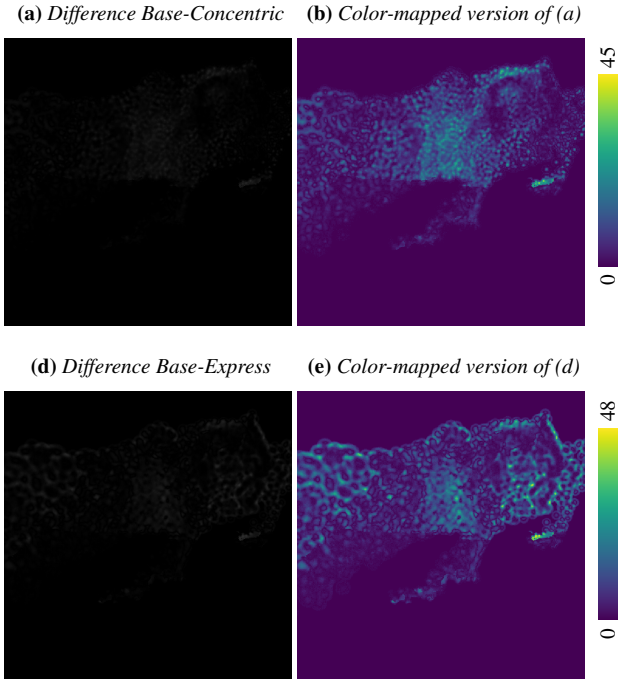
6. Conclusion

We presented a method for visualizing complex point clouds where a different quality value is assigned to each sample. The points with



(i) *Splatting* **(j)** *Base* **(k)** *Concentric* **(l)** *Express*

Figure 6: *Comparison of our non-uniform density sphere models. The top row shows a rendering of a single sphere, the middle row shows a rendering of two colored intersecting spheres, while the bottom row shows a crop of the Vineyards dataset rendered with the corresponding sphere model.*



(a) *Difference Base-Concentric* **(b)** *Color-mapped version of (a)*

(d) *Difference Base-Express* **(e)** *Color-mapped version of (d)*

Figure 7: *Euclidean image differences of the renderings of Figure 6. The top row shows in (a) the difference between the Base-Model and Concentric-Model. (b) shows a color-mapped version to better highlight the differences. The bottom row shows the same difference images for the Express-Model.*

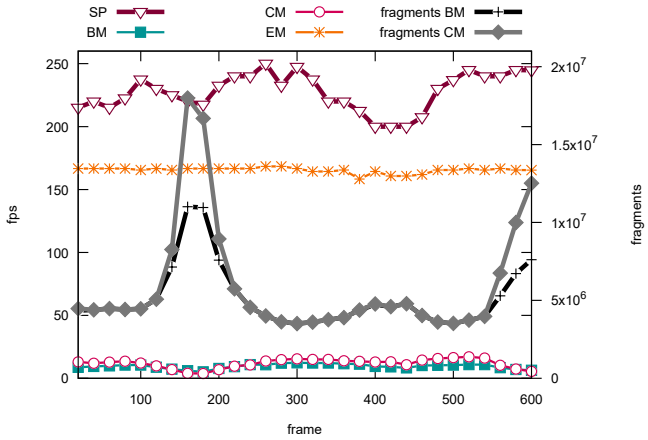


Figure 8: *Run-time in frames-per-second and total number of generated fragments plotted for a camera path on the vineyards dataset using point splatting (SP) and our method with Base-Model (BM), Concentric-Model (CM), and Express-Model (EM).*

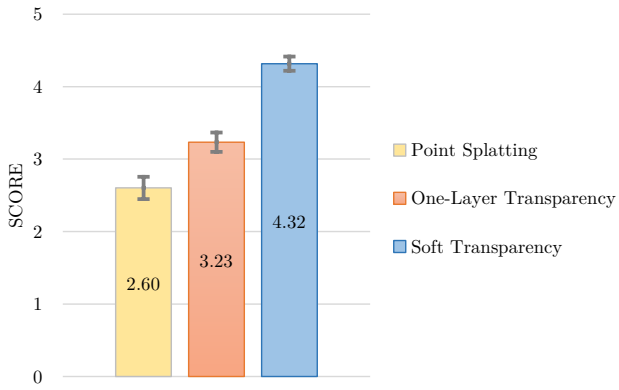


Figure 9: *User study results. For each tested algorithm the chart shows the average score with the relative 95% confidence interval.*

quality data above a certain threshold are visualized using semi-transparent spheres with non-uniform density.

The method offers better visualization of complex scenes, particularly when data of different quality (i.e. complete objects and vegetation) are mixed.

The main advantage of the proposed method is its simplicity and flexibility, that allows for high performance even with very complex datasets. We showed that from the visual point of view, our method is indistinguishable from more complex rendering models, where physical accuracy leads to poorer performances. Additionally, the degree of transparency of the elements in the scene can be changed in real-time, so that users can adapt the visualization to the goal of the point cloud inspection.

A user study, based on videos where quality encoded different characteristics of a point cloud (i.e. data uncertainty, or external and internal structures) shows that our method outperforms the currently available ones in the understanding the global structure and the elements of a scene.

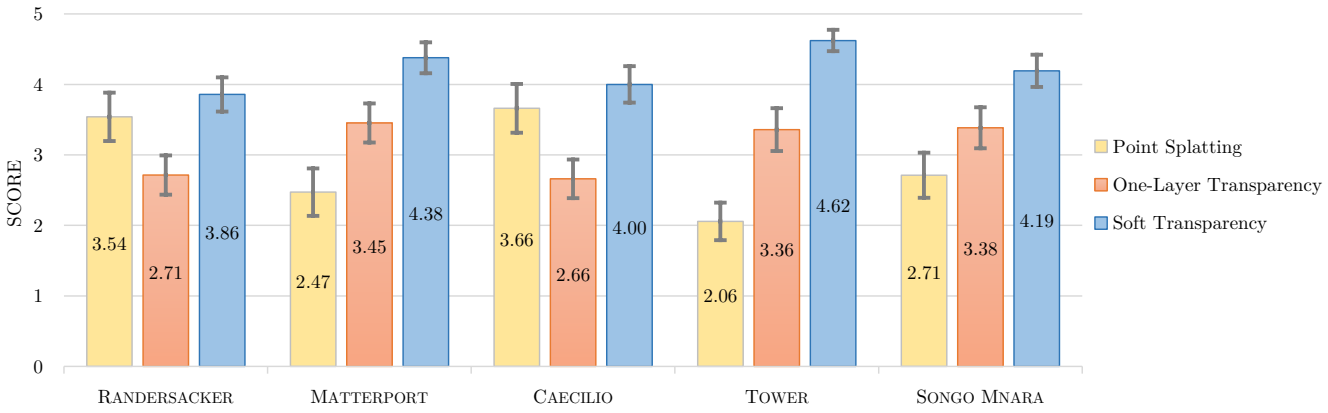


Figure 10: *User study results aggregate for each scene. For each tested algorithm the chart shows the average score with the relative* 95% *confidence interval.*

Regarding future work a few research directions could be explored to exploit this approach in an even more fruitful way:

- Extension to multi-resolution frameworks: once we have a technique to associate the quality values of intermediate resolution in a consistent way, the method can be extended so that datasets of any size may be streamed and rendered.
- Further simplification of tunable parameters: some of them (i.e. transparency modulation) may be made adaptive so that the rendering method could fit to different scenes and points of view.
- Handling uncertainty in data, by working on specific estimation associated with the way data were acquired (i.e. 3D scanning or MVS) in order to leverage the characteristics of different types of devices or techniques.
- Different types of encoding of quality information. For example, transparency may be useful to visualize temporal data, where part of the scene geometry changed over time.

7. Acknowledgements

This work was partially financed by the SCIADRO project, co-funded by the Tuscany Region (Italy) under the Regional Implementation Programme for Underutilized Areas Fund (PAR FAS 2007-2013) and the Research Facilitation Fund (FAR) of the Ministry of Education, University and Research (MIUR). We would like to thank *Surface & Edge 3D Inc.* for providing us the laser scans of the LAND BUILDING dataset.

References

[BG07] BRUCKNER S., GRÃŰLLER E.: Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (Nov 2007), 1344–1351. 3

[BHZK05] BOTSCH M., HORNUNG A., ZWICKER M., KOBBELT L.: High-quality surface splatting on today's gpus. In *Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proceedings* (2005), IEEE, pp. 17–141. 2

[BSK04] BOTSCH M., SPERNAT M., KOBBELT L.: Phong splatting. In *Proceedings of the First Eurographics conference on Point-Based Graphics* (2004), Eurographics Association, pp. 25–32. 2, 5

[Car84] CARPENTER L.: The a -buffer, an antialiased hidden surface method. *SIGGRAPH Comput. Graph. 18*, 3 (Jan. 1984), 103–108. 2

[CDF*17] CHANG A., DAI A., FUNKHOUSER T., HALBER M., NIESSNER M., SAVVA M., SONG S., ZENG A., ZHANG Y.: Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)* (2017). 7

[CFM*13] CARNECKY R., FUCHS R., MEHL S., JANG Y., PEIKERT R.: Smart transparency for illustrative visualization of complex flow surfaces. *IEEE Transactions on Visualization and Computer Graphics 19*, 5 (May 2013), 838–851. 3

[DRL10] DOBREV P., ROSENTHAL P., LINSEN L.: Interactive image-space point cloud rendering with transparency and shadows. 2

[DWE02] DIEPSTRATEN J., WEISKOPF D., ERTL T.: Transparency in interactive technical illustrations. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 317–325. 3

[ER00] EBERT D., RHEINGANS P.: Volume illustration: Non-photorealistic rendering of volume models. In *Proceedings of the Conference on Visualization '00* (Los Alamitos, CA, USA, 2000), VIS '00, IEEE Computer Society Press, pp. 195–202. 3

[ESSL11] ENDERTON E., SINTORN E., SHIRLEY P., LUEBKE D.: Stochastic transparency. *IEEE transactions on visualization and computer graphics 17*, 8 (2011), 1036–1047. 2

[GBP06] GUENNEBAUD G., BARTHE L., PAULIN M.: Splat/Mesh Blending, Perspective Rasterization and Transparency for Point-Based Rendering. In *Symposium on Point-Based Graphics* (2006), Botsch M., Chen B., Pauly M., Zwicker M., (Eds.), The Eurographics Association. 2

[HGH*10] HUMMEL M., GARTH C., HAMANN B., HAGEN H., JOY K. I.: Iris: Illustrative rendering for integral surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1319–1328. 3

[HSO07] HARRIS M., SENGUPTA S., OWENS J. D.: Parallel prefix sum (scan) with cuda. *GPU gems 3*, 39 (2007), 851–876. 6

[JB10] JANSEN J., BAVOIL L.: Fourier opacity mapping. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games* (2010), ACM, pp. 165–172. 2

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (June 1995), 99–108. 3

[MCTB12] MAULE M., COMBA J. L., TORCHELSEN R., BASTOS R.: Memory-efficient order-independent transparency with dynamic fragment buffer. In *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on* (2012), IEEE, pp. 134–141. 2, 5

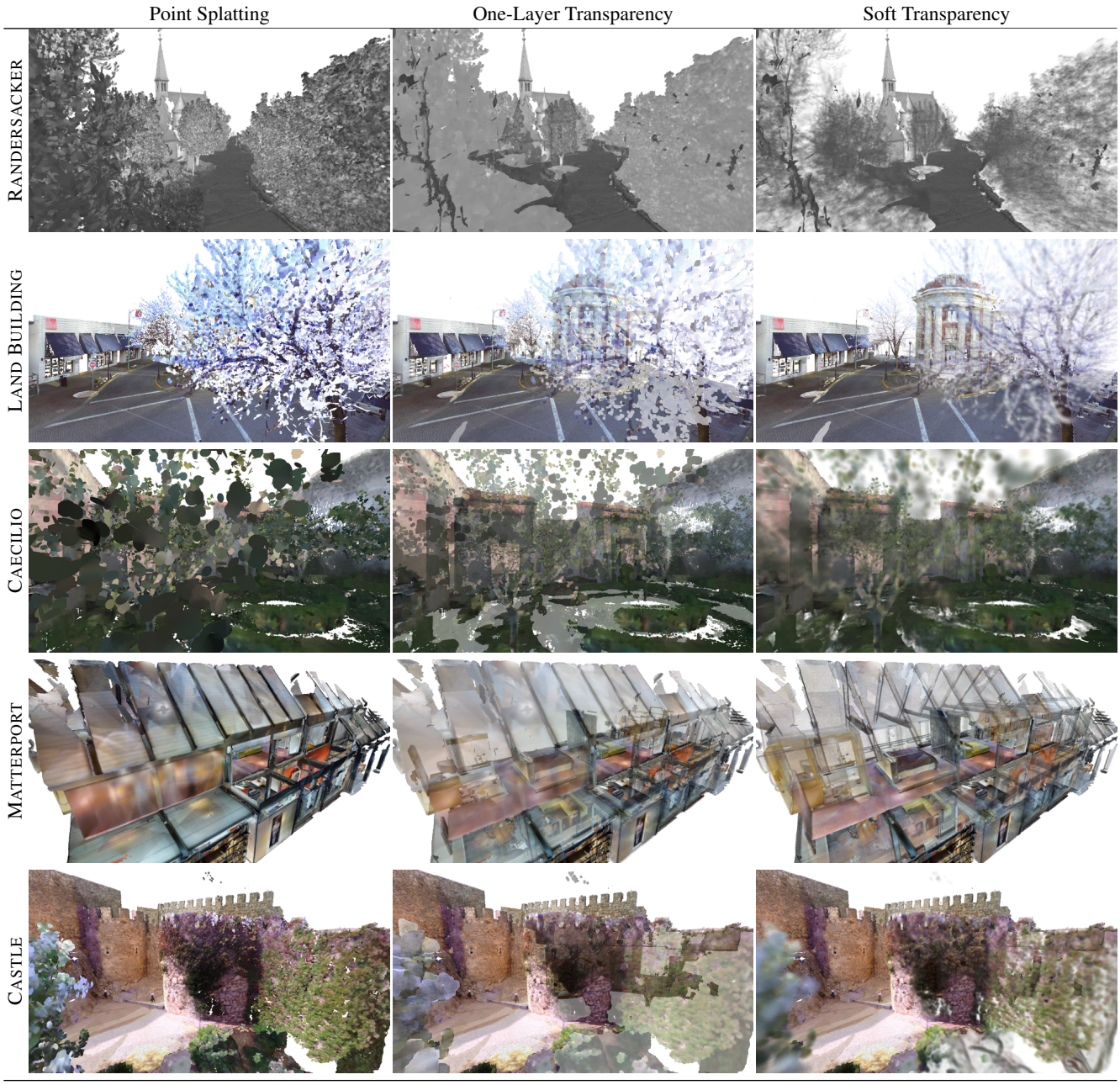


Figure 11: *Overview of our datasets (1/2). The first column shows the regular surface splatting rendering. The second column show the rendering of the transparent points by assigning a user-specified transparency value to the first layer of the splat rendering. The last column shows our Soft Transparency method using the Express-Model.*

[MCTB13] MAULE M., COMBA J. A., TORCHELSEN R., BASTOS R.: Hybrid transparency. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2013), I3D '13, ACM, pp. 103–118. 2

[PZVBG00] PFISTER H., ZWICKER M., VAN BAAR J., GROSS M.: Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 335–342. 2

[SEA09] SVAKHINE N. A., EBERT D. S., ANDREWS W. M.: Illustration-inspired depth enhanced volumetric medical visualization. *IEEE Transactions on Visualization and Computer Graphics 15*, 1 (Jan. 2009), 77–86. 3

[SGG15] STAIB J., GROTTEL S., GUMHOLD S.: Visualization of particle-based data with transparency and ambient occlusion. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 151–160. 3, 4

[SGM*11] SCHOTT M., GROSSET A. P., MARTIN T., PEGORARO V., SMITH S. T., HANSEN C. D.: Depth of field effects for interactive direct

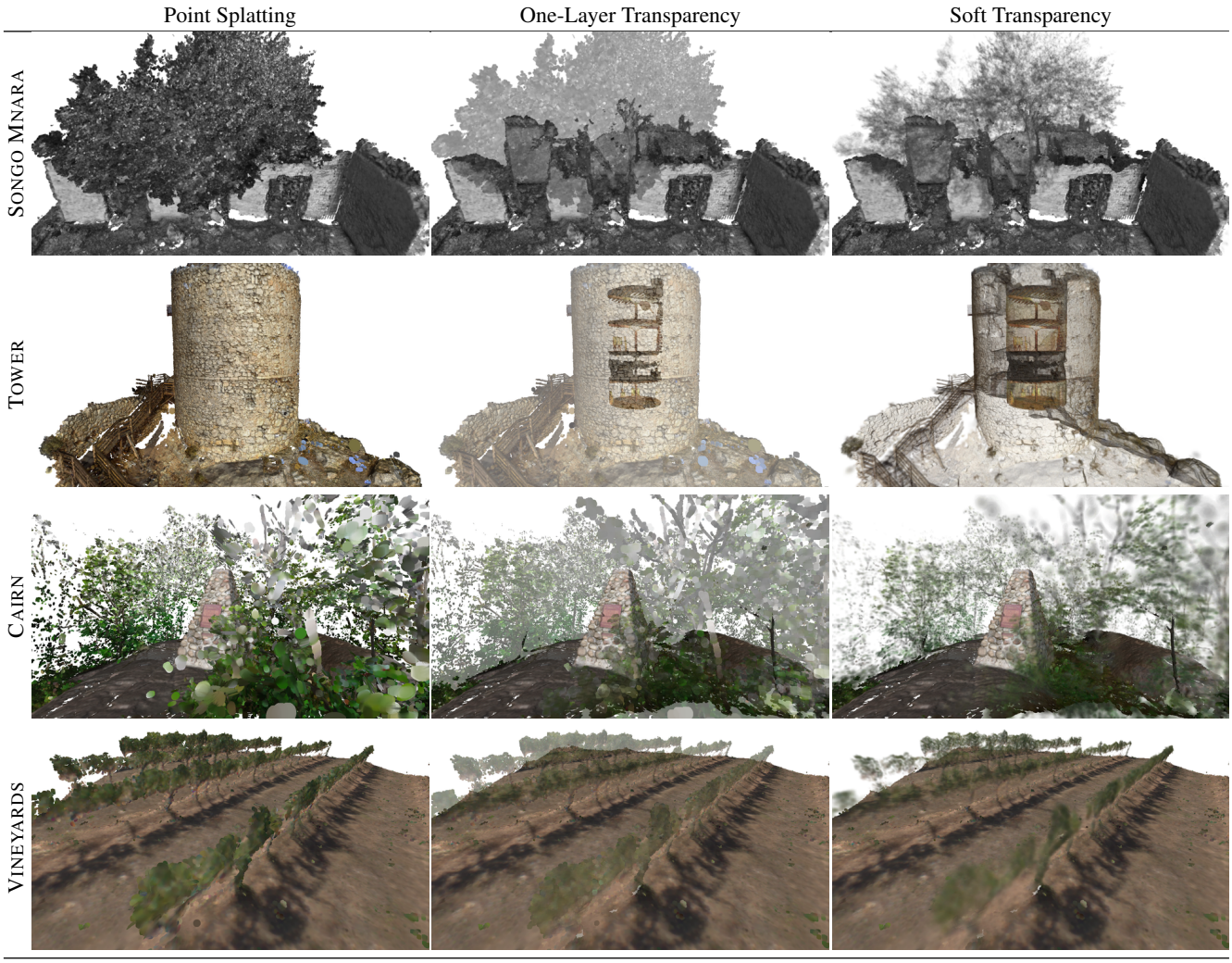


Figure 12: *Overview of our datasets (2/2). The first column shows the regular surface splatting rendering. The second column show the rendering of the transparent points by assigning a user-specified transparency value to the first layer of the splat rendering. The last column shows our Soft Transparency method using the Express-Model.*

volume rendering. *Computer Graphics Forum 30*, 3 (2011), 941–950. 3

[SML11] Salvi M., Montgomery J., Lefohn A.: Adaptive transparency. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics* (2011), ACM, pp. 119–126. 2

[SPCS16] Sabbadin M., Palma G., Cignoni P., Scopigno R.: Multi-View Ambient Occlusion for Enhancing Visualization of Raw Scanning Data. In *Eurographics Workshop on Graphics and Cultural Heritage* (2016), Catalano C. E., Luca L. D., (Eds.), The Eurographics Association. 7

[TUH*14] Tanaka S., Uemura M., Hasegawa K., Kitagawa T., Yoshida T., Sugiyama A., Tanaka H. T., Okamoto A., Sakamoto N., Koyamada K.: Application of stochastic point-based rendering to transparent visualization of large-scale laser-scanned data of 3d cultural assets. In *2014 IEEE Pacific Visualization Symposium* (March 2014), pp. 267–271. 3

[VF12] Vasilakis A., Fudos I.: S-buffer: Sparsity-aware multi-fragment rendering. In *Eurographics (short papers)* (2012), Citeseer, pp. 101–104. 2

[YHGT10] Yang J. C., Hensley J., Grün H., Thibieroz N.: Real-time concurrent linked list construction on the gpu. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 1297–1304. 2

[ZP06] Zhang Y., Pajarola R.: Single-Pass Point Rendering and Transparent Shading. In *Symposium on Point-Based Graphics* (2006), Botsch M., Chen B., Pauly M., Zwicker M., (Eds.), The Eurographics Association. 3

[ZPVBG01] Zwicker M., Pfister H., Van Baar J., Gross M.: Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 371–378. 2

[ZRB*04] Zwicker M., Räsänen J., Botsch M., Dachsbacher C., Pauly M.: Perspective accurate splatting. In *Proceedings of Graphics interface 2004* (2004), Canadian Human-Computer Communications Society, pp. 247–254. 2