Vis Comput
DOI 10.1007/s00371-017-1420-7

ORIGINAL ARTICLE

Stroke-based splatting: an efficient multi-resolution point cloud visualization technique

Rafael Kuffner dos Anjos[1,2] · **Claudia Sofia Ribeiro**[1] · **Daniel Simões Lopes**[2] · **João Madeiras Pereira**[2]

Abstract Current state-of-the-art point cloud visualization techniques have shortcomings when dealing with sparse and less accurate data or close-up interactions. In this paper, we present a visualization technique called stroke-based splatting, which applies concepts of stroke-based rendering to surface-aligned splatting, allowing for better shape perception at lower resolutions and close-ups. We create a painterly depiction of the data with an impressionistic aesthetic, which is a metaphor the user is culturally trained to recognize, thus attributing higher quality to the visualization. This is achieved by shaping each object-aligned splat as a brush stroke, and orienting it according to globally coherent tangent vectors from the Householder formula, creating a painterly depiction of the scanned cloud. Each splat is sized according to a color-based clustering analysis of the data, ensuring the consistency of brush strokes within neighborhood areas. By controlling brush shape generation parameters and blending factors between neighboring splats, the user is able to simulate different painting styles in real time. We have tested our method with data sets captured by commodity laser scanners as well as publicly available high-resolution point clouds, both having highly interactive frame rates in all cases. In addition, a user study was conducted comparing our approach to state-of-the-art point cloud visualization techniques. Users considered stroke-based splatting a valuable technique as it provides a higher or similar visual quality to current approaches.

Keywords Point cloud visualization · Non-photorealistic rendering · Splatting · Householder formula

1 Introduction

The relevance of point cloud data has recently increased with the introduction of cheaper and ubiquitous depth sensing solutions (e.g., Microsoft Kinect, Asus Xtion or PMD CamBoard) or user applications (e.g., Autodesk 123D Catch and Google's Project Tango). Creating a three-dimensional virtual representation of the real world is as accessible as ever, although at lower resolutions compared to expensive commercial solutions that produce high-resolution scans. Novel interaction paradigms [8,11] also increased the requirements for visualization of such data, where current rendering techniques were not suitable.

Point cloud visualization is a challenging field due to the unstructured nature of the data and its sparsity. Typical mesh reconstructions can be a time-consuming task. Splats have shown to have a comparable visual appearance to closed surfaces for visualization uses [4,31]. However, this is not verified when visualizing a high-resolution scan at a given distance, where the size of each splat is bigger than

This work was supported by the European Research Council under the project (Ref. 336200). Also partially supported by national funds through FCT with reference UID/CEC/50021/2013 and IT-MEDEX PTDC/EEI-SII/6038/2014. We gratefully acknowledge the support of NVIDIA Corporation.

✉ Rafael Kuffner dos Anjos
rafael.kuffner@fcsh.unl.pt

Claudia Sofia Ribeiro
claudia.ribeiro@fcsh.unl.pt

Daniel Simões Lopes
daniel.lopes@inesc-id.pt

João Madeiras Pereira
jap@inesc-id.pt

1 FCSH, Universidade Nova de Lisboa, Av. de Berna 26 C, 1069-061 Lisbon, Portugal

2 INESC-ID Lisboa, Rua Alves Redol 9, 1000-029 Lisbon, Portugal

Published online: 18 July 2017

a small region of pixels. This is a common and undesired feature when using low-resolution scans or when close-ups are part of the interaction. The shape and colors of each individual splat closely resemble pixelized artifacts from a low-resolution image.

This paper presents stroke-based splatting, an alternative visualization technique which applies concepts from stroke-based rendering to surface-aligned splatting (SAS). We explore an analogy between art, namely the fin de siécle Impressionist movement, and interactive visualization technology. Since users are culturally trained to interpret this painting metaphor, a higher quality is attributed to point clouds rendered in this painterly fashion, than to a splatted volume, or a blurred mesh on a close-up. We conducted a user study that corroborates this claim and also verifies that our approach is perceived as owning a higher quality than state-of-the-art surface-aligned elliptical splats [3,28] in all scenarios. A similar approach, the image-space non-photorealistic rendering (ISNPR), has been successfully used in the past to visualize scanned volumes [39] on an architectural context. However, this class of techniques will suffer from aspects inherent to a more flexible 3D interaction or noisier data, as seen in Sect. 5.1.

When compared to state-of-the-art point cloud visualization techniques (e.g., surface-aligned elliptical splats), our work contributes to: (1) application of splats using a real-world metaphor which allows for richer visualizations; (2) splat shape in the form of brush strokes that allow a clearer perception of the curvature of the object, generated in real time by a novel lightweight technique; (3) orient each splat more accurately, through the application of the Householder formula, informing the user on the underlying shape of the object; (4) implement an alternative optical splat blending technique which creates smooth surfaces without compromising the interactivity of the system; (5) visualize different size laser-scanned points datasets with less noise; and (6) estimate efficiently tangent vectors leading to higher interactive frame rates.

2 Related work

Surface reconstruction has been the standard approach to visualize point cloud data [10]. Several authors have developed successful techniques that estimate the original surfaces from point sets [13,19], which are capable of estimating missing data from a faulty reconstruction [6] or successfully dealing with noisy input [21]. However, the quality of the reconstruction still lingers on the resolution of the input cloud. Detail on a surface can still be wrongfully interpreted as noise in order to create a smooth output. Given the fact that point clouds naturally require less storage information than triangular meshes, it is still more efficient to use points for visualization purposes.

Several point-based rendering techniques were proposed that vary according to the primitive chosen to render a point sample as reported by Rusu and Cousins [34]. Rendering point clouds with point primitives has several drawbacks when compared to other techniques (e.g., background/foreground confusion, loss of definition on close-ups), when confronting a low-resolution scenario [1]. Katz et al. [17] solved the problem of background foreground confusion by estimating the direct visibility of sets of points. A similar goal was shared by Awano et al. [2] for organized point clouds.

Screen-aligned splats [38] have been proposed as a more efficient alternative to polygonal mesh rendering [31]. More recent solutions combined both approaches [18] to achieve a more efficient rendering of three-dimensional models keeping the quality compromise. Other techniques will align splats to an estimated surface of the object [3,28,43], which creates a better approximation of the surface. Blending between neighboring patches has been implemented through Gaussian blur [28], alpha normalization [27], and weighted averages between neighboring patches [29].

Nevertheless, the same visual fidelity edges of each individual splat are clearly visible in a close-up interaction, and compromise the quality of the visualized data. Regarding the content or shape of the rendered splats it has not been thoroughly explored in order to improve user experience on close-ups.

Non-photorealistic rendering (NPR) techniques are normally applied as post processing effects on a two-dimensional image space, applying brush strokes at a pixel level [12,26]. A wide variety of expressive styles [7,16,25] have been successfully applied to convert images in artistic styles such as Impressionism or Pointillism [41]. The typical NPR scenario aims to process a 2D image to create a stylized output. Several image aspects are analyzed in order to create more adequate brush strokes or textures applied to each part of the image. Using this class of NPR as a tool for point cloud, visualization has been explored by Xu et al. [39,40]. Authors also claim that a stylized visualization is sometimes preferred by architects when visualizing buildings, which is also achieved by their NPR technique. Normal vectors and other point information are used to detect feature points instead of image features in a first step, and then brush strokes are applied in the rendered 2D image space according to the found 3D features. While this approach might be appropriate to create stylized renderings of images, it has shortcomings when applied to point cloud visualization (Sect. 5.1).

A different paradigm, which is closer to splat rendering and our proposed approach, consists of applying brush strokes or other rendering primitives in a 3D space. Previous work from Runions et al. render point sets as semi-connected ribbons [30], which is a non-photorealistic representation that can be viable in some scenarios. The best application

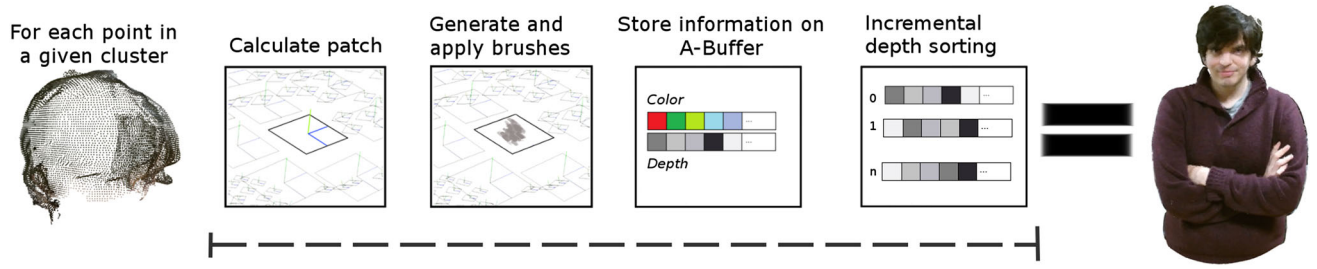


Fig. 1 Overview of the rendering step

of this concept comes from an interactive authoring system where the user manually applies brush strokes to a three-dimensional surface, called Overcoat [35]. It enforces the idea that using a 3D brush stroke-based technique to render automatically volumes is a viable and promising alternative visualization providing visually attractive representations of objects. This is the goal of our proposed approach.

3 Stroke-based splatting on point clouds

3.1 Overview

With laser-scanned data, it can be assumed that the points within a close vicinity are equally spaced up to an error value ϵ. By calculating local surface-aligned splats sized $r + \epsilon$, r being the estimated local resolution of points, we can use each splat as a small canvas for our stroke-based technique. These are created according to the estimated normal [32] together with associated tangent and bitangent vectors [23]. Here, we render brush strokes with different properties according to each estimated splat, and apply one of two different blending processes. The resulting "cloud of brush strokes" creates a painting-like visualization of the data. Ultimately, our goal is to provide high visual acuity of every element of the point cloud at a low cost. However, through manipulating the rendering parameters, the user is able to make different aesthetic choices in order to better fit the application in hand, or simulate a certain painting style.

Our process is divided into two stages: a preprocessing step where parameters are extracted from the input point cloud, and the multi-pass rendering step where the stroke-based splatting technique will be applied (Fig. 1).

Our technique was implemented using OpenGL and GLSL, version 4.5, on an interactive system that allows for free visualization around the captured point cloud. For the preprocessing stage, most of the point cloud manipulation was performed using the publicly available Point Cloud Library [33].

3.2 Preprocessing

A short offline preprocessing step is required to estimate surface parameters that will be used in the rendering step. We aim to maintain brush coherence within each section of the painting, as it is customary in oil painting. This is achieved by taking into account local aspects of the data when defining splat size, orientation, and brush texture.

3.2.1 Clustering

We use a color-based region-growing clustering algorithm as described by Zhan et al. [42] to group points with positional and color similarities. Clustering parameters can be fine-tuned to have a more fine-grained segmentation or not. Figure 2a shows different clustering settings applied to a real-world scanned input. Each splat is sized $r + \epsilon$ where r is the local resolution estimated by the average distance in the K nearest neighbors of a given point. The error value ϵ can be estimated by the standard deviation of this distance, or chosen by the user to control the brush size according to the shape of that cluster. Consequentially, denser regions use smaller brushes, while background data make use of larger brushes. Figure 2c shows an example of two different sections of a scan with different resolutions, and the calculated splats according to the necessary size.

The size of the detected cluster is also used to define brush stroke parameters. Small clusters that represent a detailed color region apply a two-layered brush stroke. Although no extra points are added to the dataset, the obtained result gives the impression of a higher resolution as seen in Fig. 2b. Other aspects such as the different proportions on each axis of the cluster or the median color can be used to apply different brush stroke generation parameters to each one of the clusters, in order to have a more faithful representation of a chosen painting style, or simply better suited brush strokes to the data in hand.

3.2.2 Normal estimation

Also in the preprocessing step, normal vectors are estimated by taking the point neighborhood so splats can be aligned to the object at each point. The problem of estimating normals can be solved by finding the least square fitting plane to a local surface S [32,36]. We apply this method as implemented by Rusu and Cousins in the Point Cloud Library [33] to each

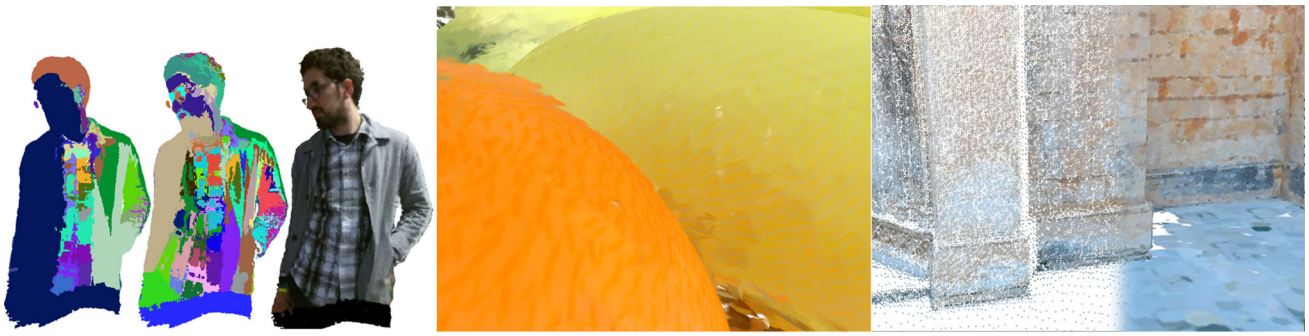


Fig. 2 Clustering approach and two different application scenarios. **a** Two different clustering settings, and the original model. **b** Small cluster on a group of fruits with two-layered painting. **c** Section of a monastery scan with clear distinction between floor and wall resolutions. Splat sizes are correctly estimated

one of the extracted clusters, searching on an area of radius $3r$, with r being the estimated cluster resolution.

3.3 Rendering

This section describes the tasks performed during the rendering step. Tangential vectors estimation through the Householder formula, brush stroke generation and blending were not calculated in preprocessing time due to the lower impact in the performance when compared to storing the results of an offline calculation.

3.3.1 Tangential vectors calculated with the Householder formula

Similarly to surface-aligned splats, we require the determination of an orthogonal reference system composed by normal, tangent and bitangent vectors. Once the normal vector is computed during preprocessing, there are several techniques to find orthogonal vectors to the given normal. We chose to use the Householder formula (HH) (Algorithm 1 after [23]), which only requires the three components of the normal vector to compute the tangent and bitangent vectors to set the splat orientation. Since this formula is deduced from collinearity and orthogonality conditions between the given normal vector and the column vectors of the Householder matrix, the resulting tangent vector formulas are, in most of its domain, continuous and smooth functions that only depend on the normal vector components.

We compared this approach to three other vector orthogonalization techniques in order to explore how well does their brush orientation emulate a natural painting flow features, such as local coherence between brush strokes: (1) the approach described in [9] where an orthonormal set is computed based on the cross-product between the given vector and the column of the identity matrix (EB); (2) the technique presented in [24] where a set of non-collinear vectors is obtained based on the analogy with a square plate mechanism (SQP) and (3) the covariance matrix (CM) technique, which has been the standard approach in surface-aligned splatting works [3]. Note that HH, EB and SQP only require the three elements of the normal vector at each point to compute the tangent and bitangent vectors (to better access the analyticity of these techniques, refer to Tables A.1, A.2, and A.3 in Appendix of [23]), whereas CM takes the neighboring points as input to define splat orientation. Consequently, the geometric properties of the tangent vectors expressed by HH, EB and SQP are non-shape aware (i.e., do not depend on point coordinates) and extrinsic (i.e., depend on the embedding Euclidean space) as an isometric transformation such as rotation applied to the point cloud affects the stroke direction. On the other hand, tangents vectors generated by CM produce shape-aware directions which are rotationally invariant. Therefore, in contrast to CM, the HH technique leads to stroke directions not determined by the shape of the point cloud, but by its embedding in space; thus, a rotated version of the point cloud would display differently directed

Algorithm 1 Pseudo-code for Householder unit vector orthogonalization (after [23])

1. Evaluate the sign of the first component, i.e., $sign(n_x)$;
2. Determine the tangent vector with the following expression:

if $n_x \geq 0$ **then**

$$\mathbf{t} = \left[-n_y \;\; 1 - \frac{{n_y}^2}{n_x+1} \;\; -\frac{n_y n_z}{n_x+1} \right]^T$$

else if $n_x < 0$ **then**

$$\mathbf{t} = \left[n_y \;\; 1 + \frac{{n_y}^2}{n_x-1} \;\; -\frac{n_y n_z}{n_x-1} \right]^T$$

end if

3. Determine the binormal vector with the following expression:

if $n_x \geq 0$ **then**

$$\mathbf{b} = \left[-n_z \;\; -\frac{n_y n_z}{n_x+1} \;\; 1 - \frac{{n_z}^2}{n_x+1} \right]^T$$

else if $n_x < 0$ **then**

$$\mathbf{b} = \left[n_z \;\; \frac{n_y n_z}{n_x-1} \;\; 1 + \frac{{n_z}^2}{n_x-1} \right]^T$$

end if

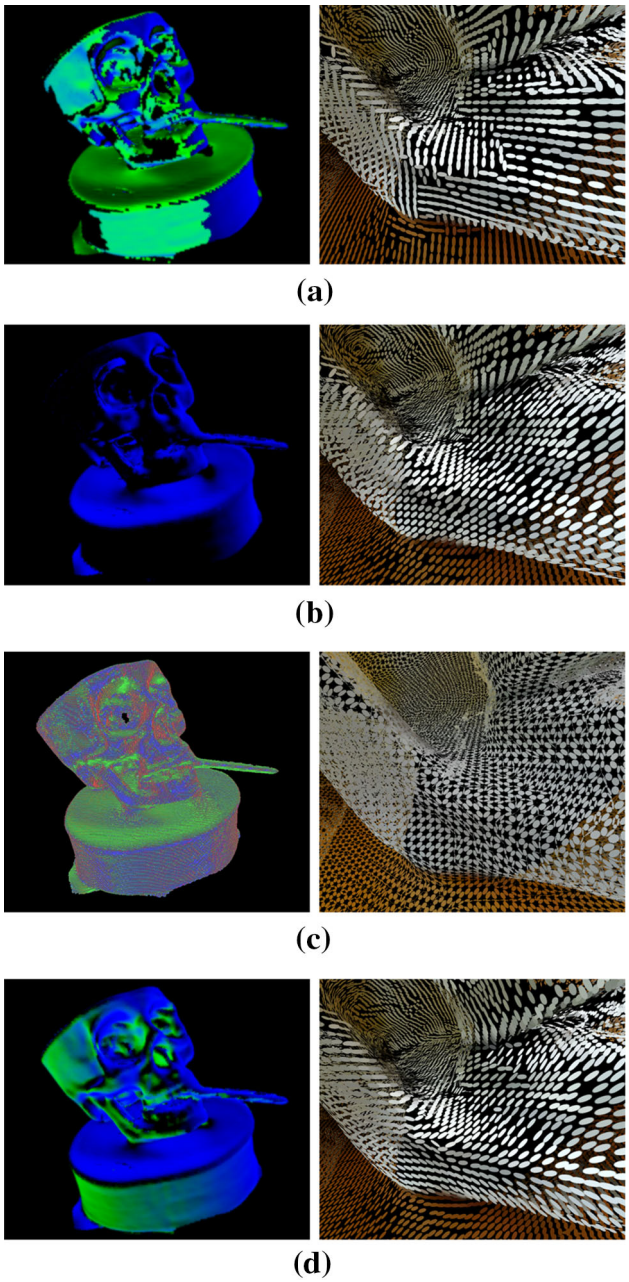


Fig. 3 Visual effect of tangential vector computation. The images reveal the differences between brush point cloud rendering using different tangent vector computation techniques. *Left column* shows direction of splats, and *right column* the final visual effect. **a** Eberly. **b** Square Plate. **c** Covariance matrix. **d** Householder

strokes. Nevertheless, this feature can be easily surpassed by computing the HH tangent vectors before rotating the vector field according to the targeted orientation for the point cloud.

Figure 3 shows the results for each one of the techniques with the top image indicating the value for the tangent vector, and the corresponding result at the bottom. The results for EB (Fig. 3a) show that harsh variations in tangent directions are found in close neighborhoods where similar normal values are to be expected, causing neighboring brush strokes to not follow the expected painting flow. SQP created more continuous tangents (Fig. 3b), but having a globally similar diagonal direction. These are expected results since EB and SQP do not provide a direct mathematical formula for the desired vector base. Such techniques rather consist of geometric processes involving testing for potential singularities and malformed vectors. Given its analytical nature, the HH formula generates tangential vectors that explicitly depend on the surface normal, thus providing the continuity control necessary to represent locally and globally consistent splat orientations.

Both the CM approach and the HH formula generate rich tangent maps, as illustrated in Fig. 3c, d. The disadvantage of using the CM in this scenario is that tangent vectors are not consistent in a close neighborhood, where slight normal vector variations present a larger effect in the calculated tangents. Although mathematically correct, it does not simulate the painting flow we aimed to achieve with this step. We found that the HH formula provided an excellent approximation to this method, while having a higher local continuity. Assuming that the normal vectors are already estimated, EB, SQP, and outstandingly CM are computationally more expensive than the HH technique.

The HH formula calculations are performed on the geometry shader, since it is more effective to calculate tangents in real time than to store three extra vertices for each point in the dataset. Such calculations are not computational heavy, and on large datasets the impact of extra storage would be far more noticeable.

3.3.2 Brush stroke generation

Previous stroke-based rendering (SBR) techniques in ISNPR typically resort on the modification of a base brush texture according to image features or user input [14,15,22,37] or fitting to an estimated spline [16]. Putting together image analysis with this approach fits to ISNPR, but not to our splatting algorithm as will be discussed in Sect. 5.1. Also, image features are computationally heavy to estimate in real time, and using the same texture for every brush stroke becomes a problem in a close-up inspection (see Sect. 5.2)

We define the shape of the brush texture according to a mathematical model based on a combination of Gaussian functions. We generate brush stroke textures with transparency which are applied to each surface-aligned splat. The user can configure a set of parameters which are commonly used in SBR techniques, in order to define the base style of brush strokes to be used. Each Gaussian function is represented by the following equation where the resulting z is the aplha value in the RGBA color space:

$$z = Ae^{-\frac{1}{2}\left(\frac{greenx_\theta - x_0}{\sigma_x}\right)^\gamma} e^{-\frac{1}{2}\left(\frac{greeny_\theta - y_0}{\sigma_x}\right)^\gamma}$$

where,

$$x_\theta = \cos\theta(u - x_0) - \sin\theta(v - y_0) + x_0$$
$$y_\theta = \sin\theta(u - x_0) + \cos\theta(v - y_0) + y_0 \quad (1)$$

and $P = (u, v)$ is the texture coordinates of the fragment being evaluated are the free variables on this equation; A is the amplitude of the Gaussian function, which can be used to control the overall opacity of the resulting texture; $P_0 = (x_0, y_0)$ is the center of the Gaussian function on the texture space coordinates; σ_x and σ_y are the Gaussian function decay according to the x and y axis; λ is the Gaussian shape exponent which controls the roundness of the function; and θ is the orientation of the function in texture space coordinates.

These values (except θ) are controlled by the user and applied globally to the point cloud, but as mentioned in Sect. 3.2.1, these are modified by fixed parameters according to the shape of the detected cluster. An example is adjusting σ_x, σ_y to make longer brushes on longer clusters, or lower γ and similar σ values for rounder brushes.

Variation between neighboring patches is introduced through a noise function with the point coordinates and normals n parameters, applied to the θ value and bounded by user defined values. Figure 4 shows an example of how this variation is introduced. From left to right each column of Gaussians share the same θ value, and subsequent columns have a θ that may not differ more than a fixed value from the previous column, in order not to create an unnatural stroking direction.

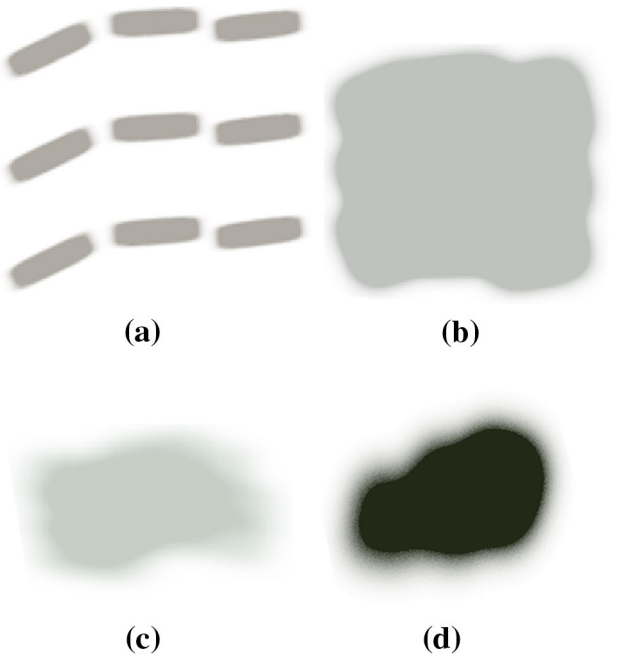


Fig. 4 Examples of brush strokes generated automatically using nine Gaussian functions as seen in **a**. **b** $A = 8$, $dev = 0.03$, $\lambda = 3$, $\sigma_x = 0.114$ and $\sigma_y = 0.0989$, **c** $A = 1$, $dev = 0.38$, $\lambda = 2.7$, $\sigma_x = 0.142$ and $\sigma_y = 0.071$ and **d** with 6 gaussians. $A = 2.27$, $dev = 0.3$, $\lambda = 2.4$, $\sigma_x = 0.0989999$ and $\sigma_y = 0.1259$

Multiple Gaussian functions allow us to correctly simulate different effects of a paint brush such as curvature in the painting direction, brush filaments, and uneven concentration of paint. The number and disposition of Gaussian functions is a user chosen parameter. The Gaussians are typically placed in a grid-like disposition, and the user indirectly defines the various P_0 values by controlling the variable dev, which defines the separation between centers. We found that we are already able to simulate widely different brush styles using between 6 to 9 Gaussians on a 3×3 grid, with less having limited expressiveness, and more having diminishing contributions on our scenario. We are also able to simulate two layers of painting by doubling these numbers and generating two overlapped textures with slightly altered color parameters. This was found to have no impact in the performance.

Similarly to splat orientation estimation, this process is performed in real time and was implemented using the geometry and fragment shaders. Brush parameters are calculated when processing each point in the geometry shader, and passed to the fragment shader, where the user defined values are also taken into account to paint the current splat with the chosen brush shape, and store information for blending.

3.3.3 Blending

Regardless of the user wanting to simulate a certain artistic style or not, blending of neighboring patches was implemented in our system to remove undesired high-frequency noise on the rendered images, located on boundaries of patches. These visual artifacts can be seen on ordinary splatting techniques, where the edge of each patch can be easily noticed. This is something that is not as apparent in the real world, which is evidently continuous. Previously, Phong shading, [3], global alpha normalization [27] and local averaging [29] were proposed to perform this step. Phong shading will be discussed ahead in Sect. 5.2. The other two proposed techniques use the alpha component to perform blending, but do not fit our proposed aesthetic, since averaging will not allow us to perceive the shape of individual splats, thus not revealing the curvature of the objects that is displayed through the individual splats.

As stated on the work from Hertzmann [16], controlling Gaussian blur allows one to eliminate undesired high-frequency noise. The downside of this approach when applied to our stroke-based rendering technique, is that blurring has little effect where each individual brush occupies a bigger group of pixels (low-resolution or close-up, Fig. 5b), which

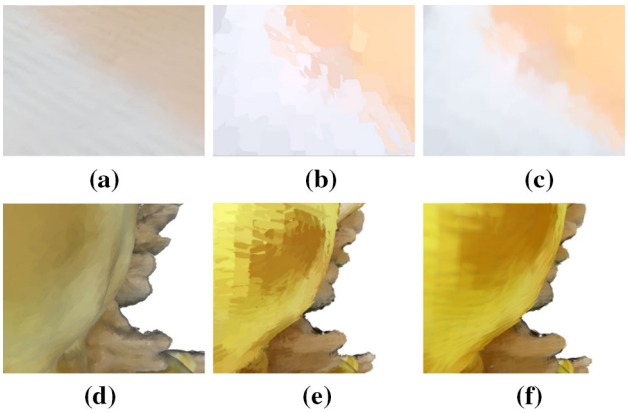


Fig. 5 Three different blending techniques experimented. A-buffer better simulated the painterly effect, while creating a smooth surface. **a** Shading. **b** Gaussian blur. **c** A-Buffer. **d** Shading 2. **e** Gaussian blur 2. **f** A-Buffer 2

is the scenario where we want to improve visualization quality. Although, when applied to a moving object, or during camera navigation, this technique fills its purpose. A two-pass weighted Gaussian blur was implemented in our system for these two situations.

We implemented an alternative blending technique which relies on the transparency component of the generated brush strokes. This is controlled by adjusting the amplitude, σ_x and σ_y values of the Gaussians. High-frequency noise is removed even with high amplitude values and low sigma values (opaque brush), since the falloff is not instantaneous due to the continuity of the Gaussian function. By applying common alpha blending on the rendered patches (stronger influence for the topmost patches), we are able to closely simulate the process of tint blending. Figure 5 shows an example of a boundary region between two colors, and the different techniques applied.

Alpha blending was implemented through the A-Buffer algorithm by using texture memory on the graphics card. Due to the high complexity of this process which would compromise the interactive aspect of the system, we opted for an incremental implementation of the bubble-sort algorithm, where its execution is divided across frames. We chose this specific sorting technique since each iteration of this technique orders one new fragment. Each new sorted fragment is used to calculate the alpha blended color and stored in the last position of the fragment array, simplifying the color calculation operation to a simple mixing of two colors in each subsequent frame. Algorithm 2 describes the technique in a high-level pseudo-code.

The sorted data are only used after a preset amount of iterations, ensuring a minimum amount of fragments to calculate a blended color of neighboring patches. Gaussian blur blending with Z-buffer occlusion is performed in the meantime. Each iteration of bubble-sort operation will enhance

Algorithm 2 Pseudo-code for A-buffering technique

```
for all pix_uv do
  if all frags sorted then
    c_res = aBuffer[n]
  else
    n_sorted = bubbleSortStep(n_total, n_sortedPrev)
    c = alphaBlendStep(n_total, n_sortedRes, n_sorted)
    aBuffer[n_sortedRes − 1] = c
    c_res = c
  end if
end for
```

the quality of our visualized data, with low impact to the interactivity of the system.

4 User study

In order to compare our approach with other existing approaches, a user study was conducted with the aim of understanding if stroke-based splatting had better quality in terms of resolution, surface and shape representation when compared to its alternatives. With this aim in mind, eleven different datasets were used to generate the images representative of each rendering approaches, specifically, mesh reconstruction, surface-aligned splats and our approach. For each technique, different datasets were rendered with different resolutions, shape complexity and color richness.

The study consisted of an online questionnaire composed by eleven 6-point Likert scale questions. Each question had two images, one for either mesh reconstruction or splatting and the other for our approach. The position of each image in each question was randomized in order not to influence/guide the participant in an aesthetic choice. Thirty-one participants filled out the questionnaire anonymously, completing the required questions. We asked each participant *"to look closely at each image and comparatively analyze which do you think has better quality in terms of resolution, surface and shape representation."*

A summary of the obtained results including descriptive as well as inferential statistics is presented in Table 1. Since our sample did not follow a normal distribution, the Wilcoxon signed-rank test was used to assess the statistical significance when comparing the results between approaches. A thorough analysis of the results and their implications of the derived conclusions of this work are presented in Sect. 5.3.

5 Results and discussion

Our algorithm was tested on a Intel i7-6700MQ at 3.40GHz desktop computer with 16,0GB and an NVIDIA Tesla K40c paired with a NVIDIA GeForce GTX 980 Ti graphics adapter. The datasets were captured by using the Microsoft Kinect

Table 1 User tests results: Median (Interquartile Range)

Dataset	SBR	Mesh	SBR (%)	Mesh (%)	Equal (%)
Giraffe	3 (2,5)	3 (1)	45	55	3
Yoda	4 (2,5)	4 (2)	48	39	13
Shirt	4 (2)	3 (1)	61	29	10
Man*	5 (2)	5 (1)	13	39	42
	SBR	Splats	SBR (%)	Splats (%)	Equal (%)
Frog	4 (1)	4 (2)	68	32	0
Girl*	5 (1,5)	3 (2)	90	3	6
Monastery*	5 (1,5)	4 (3)	71	23	6
Skull*	4(1,5)	4 (2)	65	29	6
Veggies*	5 (1)	3 (2)	77	16	6
Blonde	4 (2)	4 (1)	52	26	19
Guitar*	4 (2)	3 (3)	45	16	39

* Indicates statistical significance. Also, percentage of user preference for each rendering technique

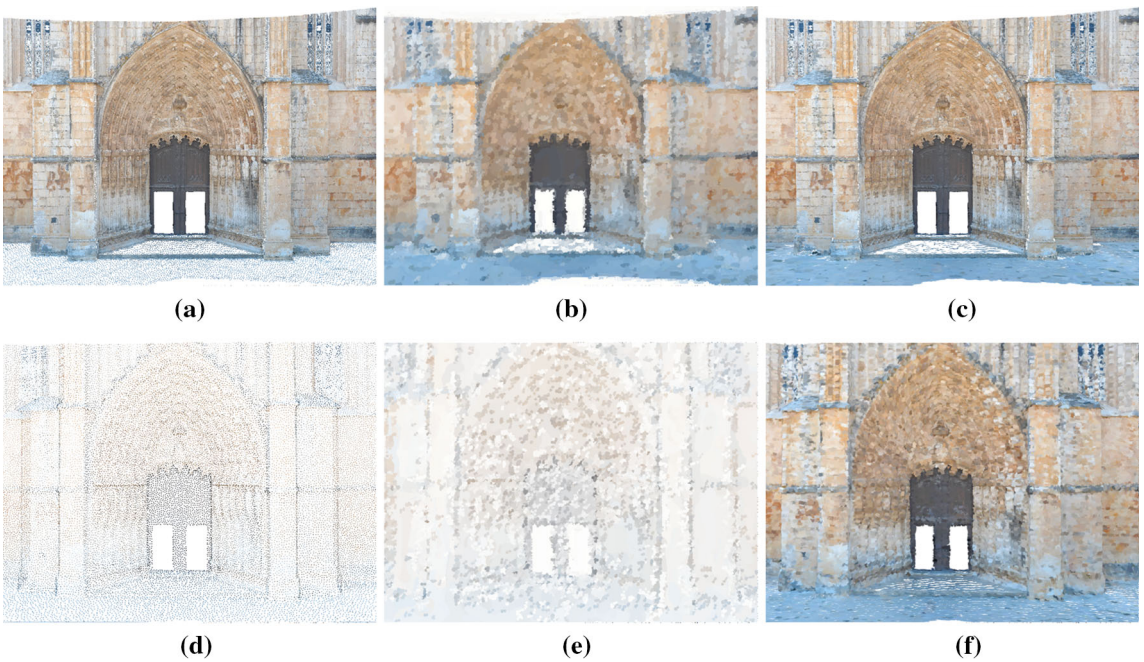


Fig. 6 Two different resolutions for the Monastery dataset under different rendering techniques: 580.062 (*top*), and 40.363 (*bottom*). Leftmost pictures contain pure point rendering with a fixed size to show the density of the dataset. **a** Point rendering 580 k. **b** Image-space NPR 580 k. **c** Our approach 580 k. **d** Point rendering 40 k. **e** Image-space NPR 40 k. **f** Our approach 40 k

One device [5], with the exception of some publicly available point clouds for benchmarking purposes. Although our main objective was to test the viability of our technique on lower resolution point clouds, we tested its scalability on very dense datasets regarding visualization and performance.

5.1 Comparison with image-space NPR

We compared our approach with an ISNPR algorithm applied to our Monastery dataset at two different resolutions as seen in the top row of Fig. 6. The most relevant works which use this approach for point cloud visualization are the ones from Xu et al. [39,40].

In a high-resolution dataset viewed at a distance, our rendered result is close to reality green (Fig. 6c), with just slight differences from the point-based rendering. ISNPR will always create a stylized result (Fig. 6b), regardless of the distance of visualization, which may not be desired in a common point cloud visualization scenario.

When applied to a low-resolution dataset (Fig. 6d), which can be considered the equivalent to zooming into a certain point cloud, point rendering leaves sparse holes, which the

big brushes applied by ISNPR techniques are not capable of covering (Fig. 6e). Our approach creates a closed surface, with only at this lowered resolution (10% of the original), showing a more stylized visual representation, while maintaining coherence in a close-up interaction.

For the goal of point cloud visualization and solving the problems of background/foreground resolution and filling holes, our approach is superior to ISNPR, which can only be applied in high-resolution datasets, up until a certain distance. For the goal of creating stylized results using point clouds as input, ISNPR has greater range of created effects due to the higher flexibility in the applied brush strokes, but still has the mentioned interaction limitations. Figure 6f shows that our approach can create a stylized result similar to an ISNPR approach with only 10% of the points, and no restrictions to the interaction.

5.2 Comparison with elliptical surface-aligned splats

When comparing our approach to the state-of-the-art visualization technique for point clouds (i.e., SAS), we acknowledge three main improvements when dealing with laser-scanned point clouds. Firstly, the change between elliptical splats to brush strokes. Not only can the impressionistic aesthetic be achieved, which aggregates a higher perceived quality to our results, but curvature can be more clearly perceived than when using elliptical splats. Figure 7c, d shows the same dataset with different splat shapes oriented in the same direction. Even from a moderate distance, strokes can be easily perceived, providing better shape perception.

Secondly, color blending through alpha values was more adequate to laser-scanned data. This has been previously used through global [27] and local averaging techniques [29], with positive results. These techniques, however, aimed to always create the illusion of a single surface, not revealing the shape and orientation of each single splat, which was one of our goals. Our blending through the A-buffer is not meant to be seen as a replacement to these techniques, but it is a better fit to our painting metaphor.

Current state-of-the-art splatting techniques based on shading [3,28] are more suitable to a scenario where someone wants to replace his mesh rendering pipeline by a splatting technique for an efficiency and quality trade-off. When a mesh is the expected input, one shall have strictly defined materials for each segmented component, and Phong shading can be applied to create high-quality representations. With a laser scanner input only color values representing the real-world lighting conditions are available. The absence of well defined materials leads to an inadequate aspect for the rendered results, as seen in Fig. 7a, b.

Finally, splat orientation. CM is considered the standard method to estimate tangent vectors for splat orientation. However, we found that the HH formula presents several advantages when applied to our specific scenario. For instance, HH tangential vectors provide an interesting visual effect where "virtual paint brush strokes" flow smoothly and continuously throughout the point cloud domain, without any unpleasant rendering artifacts. Even though each point is not locally approximated by a quadric surface, the HH tangent vectors do carry first-order differential information of the estimated surface normal plane. This contributes to both local and global coherence of the splat orientations.

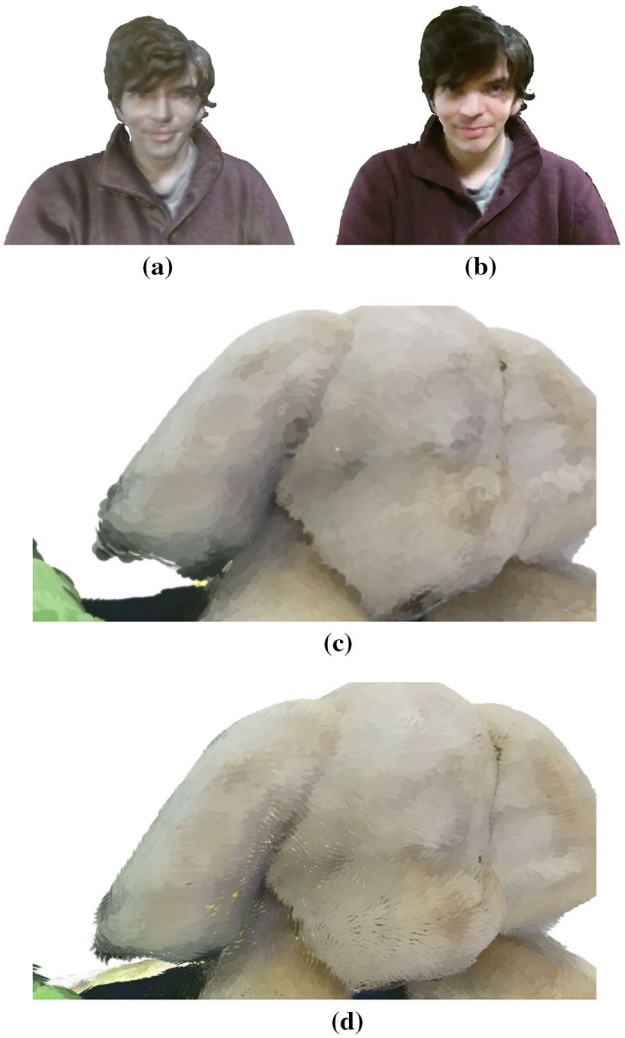


Fig. 7 In depth comparison between our technique and Surface-Aligned Splats. **a** SAS with Phong shading [3]. **b** Our approach. **c** Elliptical splats. **d** Brush strokes

Despite the richness of geometric attributes associated with the CM technique (i.e., it provides a full reference frame composed by normal, tangent and bitangent vectors oriented along the local and anisotropic density of points), solving the eigenvalue and eigenvector problems is computationally costly as it demands the calculation of the Jacobian matrix and to solve the characteristic polynomial. The number of FLOPS of the HH formula to compute both tangent

and bitangent vectors, given a unit vector, sums up to: 1 order operation, 3 summations/subtractions, 10–12 multiplications, and 1 division. This is much less demanding than solving the eigenvalue problem for a 3×3 matrix. Hence, the HH technique allows for real-time tangent computation. Note that this advantage holds under the assumption that the normal vectors have been estimated by a more efficient technique than the CM method. For this purpose, there are several approaches to estimate surface normals for point data [20], which must be chosen according to the best graph structure that represents the neighboring points.

Moreover, the HH formula holds for a wider diversity of input data types, namely depth maps, analytical representations such as implicit surfaces, and meshes or point clouds with high or low quality, while CM is only suitable for good quality meshes and point clouds.

Contrary to the CM techniques which aligns splats along the local and anisotropic density of points, the HH formula uses an isotropic radius as it does not consider point density, only the direction and sense of the local normal vector are used for tangent calculation. This can be seen as advantage whenever the point cloud is locally noisy, under sampled or in the presence of holes, which is the "scenario par excellence" of our work. As illustrated in Fig. 3c, with the covariance matrix technique, the tangent orientation is less locally coherent than the HH formula, as seen in Fig. 3d.

5.3 Visual perception of point clouds: user test results.

We compared our rendering technique to Meshes and Surface-Aligned Splats [3,28], the two most popular visualization techniques to represent three-dimensional data. Our results were validated through the presented user study (see Table 1).

Figure 8a, b and c shows three different datasets comparing our approach to a mesh. From a distance, similar to splats or point rendering, our technique was perceived as equal to a mesh. This was confirmed for our technique in the dataset "Man" on our user study, where both techniques had the same median value, and the majority (42%) of the test subjects graded both techniques equally. Datasets Giraffe and Yoda (Fig. 8b, c) which are slightly closer to the captured objects showed no significant difference between approaches. Both median values and percentage of preference are similar, indicating a similar performance from SBR to meshes.

Although no statistical significance was found on the zoomed in "Shirt" dataset results (Table 1; Fig. 8a), our approach has a higher median value and user preference, with double the amount of users grading SBR higher than a closed in mesh. We believe that these results still indicate the adequacy of SBR on closed in datasets. Meshes display blurred textures due to the loss of resolution when closing up. Our technique does not display blurred data, but individual brush strokes that resemble those in an artistic painting. This can also be noted in Fig. 8d–f. A full point cloud, and its 41.3 and 9.9% reduction have comparable visual results. As the number of points decreases, we simply approximate ourselves to a more impressionistic depiction of the rendered model, which is not necessarily perceived as a lower quality one.

Another relevant point comes from the fact that surface reconstruction techniques [1,19] lose details on lower resolution point clouds in order to obtain a smooth surface. We argue that in this scenario, our visualization offers a viable alternative to the rendering of a reconstructed surface.

When compared directly to splats, our technique was perceived to have higher quality by the majority of users in every displayed dataset, including the ones with similar median values. Figure 9 shows comparisons between surface-aligned splats [28] and our technique on the best performing scenarios.

Although less noticeable than in the screen-aligned splats approach, they still distort the more delicate shapes of objects (Fig. 9c, top). On this example where a familiar shape (human face) is displayed, our approach had the biggest edge over splats, with a difference of 2 in the median quality value, and a 90% preference by the users.

Figure 9e shows a highly detailed Monastery entrance where SBR allows for clear perception of individual arches and structures through the orientation of brushes, while circular splats (Fig. 9a) do not. Figure 9b, d shows a zoomed out data set where although no artifacts can be visibly noticed, SBR still was perceived to have a statistically significant advantage over splats. Users consistently identified and referenced our results as paintings in the user study comments, which we argue to be the reason that they were rated with higher quality.

Dataset "Skull" had similar median value results between both techniques. This is due to the fact that most of the model was white, and neither individual splats or brushes could be clearly perceived. Similar claims could be made about "Frog" and "Blonde", which have large uniformly colored surfaces. Although similar grades were given, over double the users graded our approach higher than splats in these three scenarios.

5.4 Performance

We performed tests using publicly available models from the Stanford scanning repository: Dragon (437.645 points), a dataset with three Buddha models combined (1.630.572 points) and Lucy (14.027.872 points). Due to the fact that these models do not have color information, which is necessary for patch size estimation in our technique, we included a laser scan of a Monastery with a similar dimension to the Dragon model. Interaction was always performed on highly

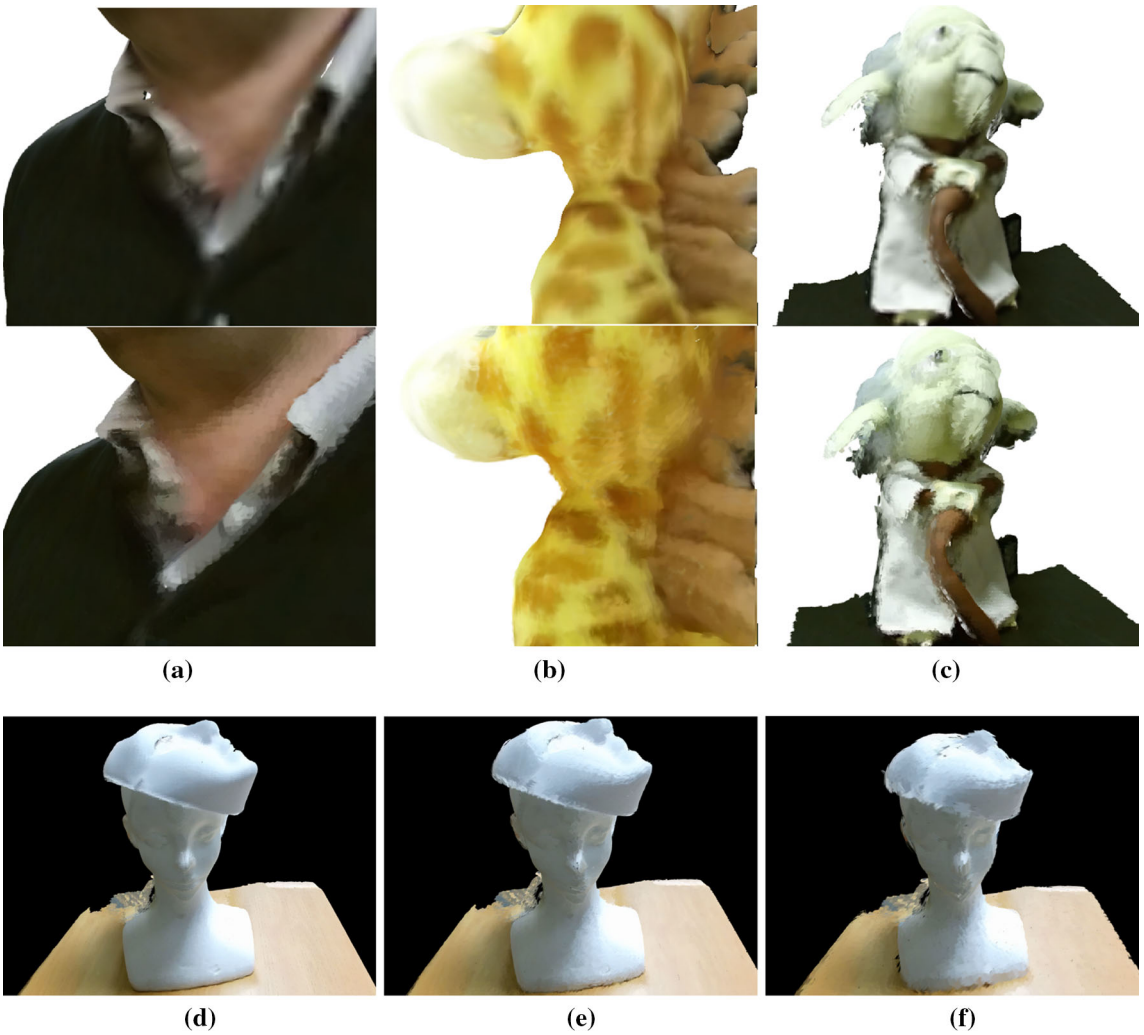


Fig. 8 Visual perception results. **a**–**c**: comparison between a unlit mesh rendering (*top*), and our technique (*bottom*). SBR has comparable quality to meshes, with a different type of resolution artifacts. While meshes blur details, SBR shows individual strokes. **d**–**f**: Same point cloud with different number of points. On a 10 times smaller cloud, rendering is closer to an artistic depiction, but still allows for accurate visual perception of shapes. **a** Shirt: 229.779 points. **b** Giraffe: 109.606 points. **c** Yoda: 34.878 points. **d** 12.295 points. **e** 50.799 points. **f** 12.221 points

interactive frame rates in a 1920×1080 window. Table 2 shows the performance results for each dataset. Median frames per second, first and third quartiles.

The complexity of the rendering activity is directly impacted by the used blending method. When blur is being used, our technique is comparable to pure point cloud rendering. The extra operations performed by our algorithm are due to the calculations of the Householder formula applied to each point to create the local patch, and the combined Gaussian function for the brush strokes at each rendered fragment. All of the necessary operations for these formulas are atomic on the graphics card where they are executed. Due to the use of clustering to determine the size of each brush stroke, the effect of triangle operations such as cutting and z buffering can be considered negligible since we aim for the least amount of overlap between neighboring patches. Blurring was found to have a negligible effect on the frame rate since the two-pass implementation is very efficient in the graphics card, minimizing the amount of texture look-ups through linear filtering.

Alpha blending operations scale with the size of the output screen times the number of layers on the A-buffer (or the limit of operations placed on the bubble-sort algorithm for each frame), and the size and shape of the chosen brush strokes. The transparency, shape and size chosen by the user determine the number of fragments at each pixel array during blending. Although these operations have a harsh effect on frame rates, seen by the higher difference between the first and third quartiles, they are only being applied when there is no interaction taking place, and for a short period of

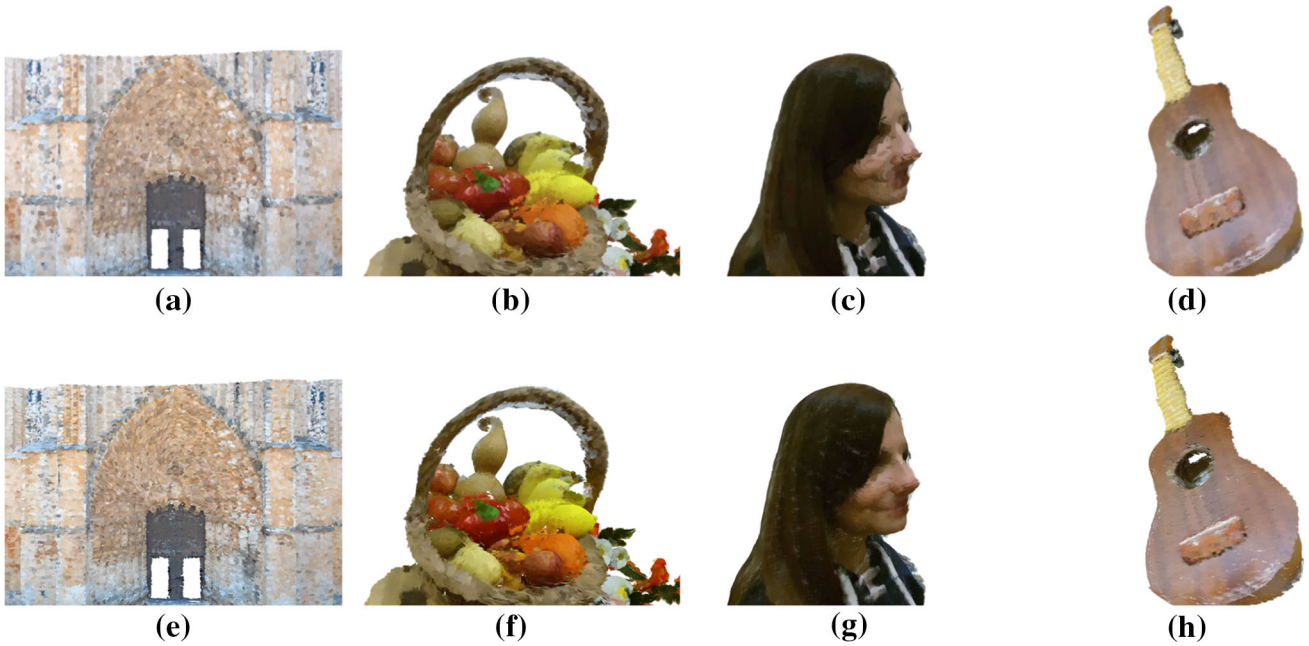


Fig. 9 Comparison between our approach (*bottom row*) and most recent surface oriented splats [28]. Biggest improvements noticed specially when fine features should be recognizable, such as **a**, **c**

Table 2 Median Frames per second (first: third quartile). Similar median values between both techniques, but higher Interquartile Range on A-buffer

Model (points)	G.Blur (fps)	A-Buffer (fps)
Dragon (566.098)	418 (368:506)	444 (302:642)
3 Buddhas (1.630.572)	254 (247:270)	257 (130:373)
Monastery (580.062)	420 (396:486)	352 (285:505)
Lucy (14.027.872)	67 (63:71)	62 (56:267)

time (first steps of the bubble-sort algorithm), hence the low effect on the median frame rates (Table 2). We found that although rendering is process intensive according to the varied mentioned factors, it does not interfere on the system's interactivity.

Given the fact that the visual acuity of a lower resolution point cloud is comparable to a denser one while using our technique, we argue that we can use largely simplified versions of such point clouds without compromising the quality of the interaction. Moreover, this allows for rendering and visualizing of larger but sparser data sets, offering a richer interactive experience.

Although the preprocessing step has computational heavy operations such as normal estimation and color-based clustering, which are bound by the KNN algorithm complexity which is $(O(n^{dk+1}logn))$, our goal was not to optimize its execution time. This is justified by the fact that these operations can be performed only once and then stored on a hard disk. Preprocessing times were found to be short on these datasets, with 10 s for Dragon, 36 s for three buddhas, one minute for Lucy, and 2 min 29 s for Monastery, due to the higher number of clusters generated. All low-resolution datasets had negligible preprocessing times.

5.5 Other results

We found that the quality and shape of the brush stroke textures greatly affected the perceived quality of the rendered result. The achieved visual aspect can closely resemble different painting styles through manipulation of the brush stroke textures blending and scale factors. Although our initial focus was not on stylizing, the artistic results achieved, especially on lower resolution data sets, open new possibilities for non-photorealistic rendering techniques (examples in the attached video for this publication). Additionally, the wide variety of brush textures we are able to generate through the combination of Gaussians can be applied in two-dimensional NPR, image processing-related fields or rendering and terrain modeling.

We also applied this technique on time-sequenced point cloud data sets (see video), where we discovered the brush attribution process to be consistent enough to prevent too much visual disturbance. Due to the fact that the varying parameters of the brush generation technique take into account the x and y coordinates of the point in question, the painting seems still on its non-animated components.

5.6 Limitations

One current limitation to our technique relates to the size of the A-buffer which needs to be allocated for proper blending. On more complex datasets, a lot of memory is inefficiently used to the limit where it might exceed what is available. Out-of-core techniques need to be considered on future work.

Secondly, fragments are sorted for each pixel, not for its original patch. If big brushes are being used, neighboring patches can intersect each other in an unnatural way. To stay truthful to our painting metaphor, a different ordering criteria should be applied, which ensures consistency in the order of application of each brush stroke.

Our system is currently targeted at scanned colored point clouds. Non-colored clouds do not work well with our segmentation algorithm, leading to patches with inadequate sizes, as noticed on the Dragon and Buddha models (as seen in the attached video for this publication). This forces us into using bigger patches in the whole model in order to have closed surfaces, but unnecessarily complicating the A-buffer sorting operation.

6 Conclusions and future work

We presented a rendering technique that is better suited to nowadays typical point cloud capture and visualization scenarios. We are able to have equal or better visual quality to the alternatives when rendering point clouds with lower point resolution, while allowing a less restrictive interaction. Stroke-based splatting was shown to have comparable results when using sub-sampled versions of the point clouds, which indicates its viability for visualizing large data sets.

Compared to splat rendering, our technique was always perceived as having a higher visual quality. We improved the direction, content, and color of each individual splat, while addressing specific problems of point clouds scanned with commodity depth sensors. When compared to meshes, we provide a stylized rendering at lower resolutions and close-ups instead of the blurred visualization typical to meshes, while maintaining visual quality at a higher point count.

A technical contribution of the paper consists of applying Householder formula to calculate both locally and globally coherent tangential vectors that are used to orient brush strokes in space. Also, a simple brush stroke generation formula was presented, allowing for generation of widely varying textures that can be used on non-photorealistic rendering techniques, image editing applications, terrain modeling, and others.

On the field of NPR, our technique is a better alternative to generating stylized visualizations in an interactive scenario, while still being capable of simulating a wide array of styles, adjusted in real time.

Future work goes through optimizing the normal estimation and clustering in order to perform real-time renderings in scenarios using a single Microsoft Kinect sensor, hence, enabling a whole new array of point cloud-based applications. Another future development will be extracting more information out of the detected clusters such as point cloud saliency values and features in order to generate better brush strokes, relying less on fine tuning by the user, and allowing us to create not only better visualizations but also more stylized results.

References

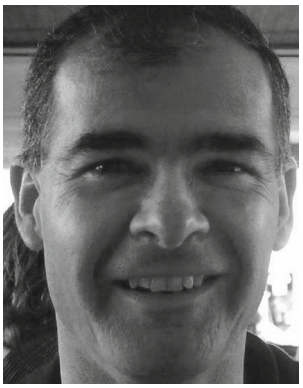
1. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Computing and rendering point set surfaces. IEEE Trans. Vis. Comput. Graph. **9**(1), 3–15 (2003). doi:10.1109/TVCG.2003.1175093
2. Awano, N., Nishio, K., Kobori, K.i.: Interactive stipple rendering for point clouds. WSCG 2010: Communication Papers Proceedings: 18th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision in co-operation with EUROGRAPHICS pp. 193–198 (2010)
3. Botsch, M., Hornung, A., Zwicker, M., Kobbelt, L.: High-Quality Surface Splatting on Today's GPUs. In: M. Alexa, S. Rusinkiewicz, M. Pauly, M. Zwicker (eds.) Eurographics Symposium on Point-Based Graphics (2005). The Eurographics Association (2005). doi:10.2312/SPBG/SPBG05/017-024
4. Botsch, M., Wiratanaya, A., Kobbelt, L.: Efficient high quality rendering of point sampled geometry. In: Proceedings of the 13th Eurographics Workshop on Rendering, EGRW '02, pp. 53–64. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2002)
5. Camplani, M., Mantecon, T., Salgado, L.: Depth-color fusion strategy for 3-d scene modeling with kinect. IEEE Trans. Cybern. **43**(6), 1560–1571 (2013). doi:10.1109/TCYB.2013.2271112
6. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3d objects with radial basis functions. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01, pp. 67–76. ACM, New York, NY, USA (2001). doi:10.1145/383259.383266
7. Curtis, C.J., Anderson, S.E., Seims, J.E., Fleischer, K.W., Salesin, D.H.: Computer-generated watercolor. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97, pp. 421–430. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1997). doi:10.1145/258734.258896
8. Dos Anjos, R.K., Pereira, J.M., Oliveira, J.F.: Collision detection on point clouds using a 2.5+ d image-based approach. J. WSCG (2012)
9. Eberly, D.: Computing orthonormal sets in 2d, 3d, and 4d. Geometric Tools. LLC (2016)

10. Fabio, R., et al.: From point cloud to surface: the modeling and visualization problem. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. **34**(5), W10–w10 (2003)
11. Goesele, M., Ackermann, J., Fuhrmann, S., Haubold, C., Klowsky, R., Steedly, D., Szeliski, R.: Ambient point clouds for view interpolation. ACM Trans. Graph. **29**(4), 95–95 (2010). doi:10.1145/1778765.1778832
12. Gooch, B., Coombe, G., Shirley, P.: Artistic vision: painterly rendering using computer vision techniques. In: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering, pp. 83–ff. ACM (2002). doi:10.1145/508543.508545
13. Gopi, M., Krishnan, S., Silva, C.T.: Surface reconstruction based on lower dimensional localized delaunay triangulation. Comput. Gr. Forum **19**, 467–478 (2000). doi:10.1111/1467-8659.00439
14. Haeberli, P.: Paint by numbers: abstract image representations. SIGGRAPH Comput. Graph. **24**(4), 207–214 (1990). doi:10.1145/97880.97902
15. Healey, C.G., Tateosian, L., Enns, J.T., Remple, M.: Perceptually based brush strokes for nonphotorealistic visualization. ACM Trans. Graph. **23**(1), 64–96 (2004). doi:10.1145/966131.966135
16. Hertzmann, A.: Painterly rendering with curved brush strokes of multiple sizes. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '98, pp. 453–460. ACM, New York, NY, USA (1998). doi:10.1145/280814.280951
17. Katz, S., Tal, A., Basri, R.: Direct visibility of point sets. ACM Trans. Graph. **26**(3) (2007). doi:10.1145/1276377.1276407
18. Kawata, H., Gouaillard, A., Kanai, T.: Interactive point-based painterly rendering. In: Proceedings of the 2004 International Conference on Cyberworlds, CW '04, pp. 293–299. IEEE Computer Society, Washington, DC, USA (2004). doi:10.1109/CW.2004.42
19. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, SGP '06, pp. 61–70. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2006)
20. Klasing, K., Althoff, D., Wollherr, D., Buss, M.: Comparison of surface normal estimation methods for range sensing applications. In: Robotics and Automation, 2009. ICRA '09. IEEE International Conference on, pp. 3206–3211 (2009). doi:10.1109/ROBOT.2009.5152493
21. Kolluri, R., Shewchuk, J.R., O'Brien, J.F.: Spectral surface reconstruction from noisy point clouds. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP '04, pp. 11–21. ACM, New York, NY, USA (2004). doi:10.1145/1057432.1057434
22. Litwinowicz, P.: Processing images and video for an impressionist effect. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97, pp. 407–414. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1997). doi:10.1145/258734.258893
23. Lopes, D.S., Silva, M.T., AmbróSio, J.A.: Tangent vectors to a 3-d surface normal: a geometric tool to find orthogonal vectors based on the householder transformation. Comput. Aided Des. **45**(3), 683–694 (2013). doi:10.1016/j.cad.2012.11.003
24. Lopes, D.S., Silva, M.T., Ambrósio, J.A., Flores, P.: A mathematical framework for rigid contact detection between quadric and superquadric surfaces. Multibody Syst. Dyn. **24**(3), 255–280 (2010). doi:10.1007/s11044-010-9220-0
25. Majumder, A., Gopi, M.: Hardware accelerated real time charcoal rendering. In: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering, pp. 59–66. ACM (2002). doi:10.1145/508539.508541
26. Meier, B.J.: Painterly rendering for animation. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96, pp. 477–484. ACM, New York, NY, USA (1996). doi:10.1145/237170.237288
27. Pajarola, R., Sainz, M., Guidotti, P.: Confetti: object-space point blending and splatting. IEEE Trans. Vis. Comput. Graph. **10**(5), 598–608 (2004). doi:10.1109/TVCG.2004.19
28. Preiner, R., Jeschke, S., Wimmer, M.: Auto splats: Dynamic point cloud visualization on the gpu. In: EGPGV, pp. 139–148 (2012)
29. Ren, L., Pfister, H., Zwicker, M.: Object space ewa surface splatting: a hardware accelerated approach to high quality point rendering. Comput. Graph. Forum **21**(3), 461–470 (2002). doi:10.1111/1467-8659.00606
30. Runions, A., Samavati, F., Prusinkiewicz, P.: Ribbons. Vis. Comput. **23**(9), 945–954 (2007). doi:10.1007/s00371-007-0153-4
31. Rusinkiewicz, S., Levoy, M.: Qsplat: A multiresolution point rendering system for large meshes. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '00, pp. 343–352. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000). doi:10.1145/344779.344940
32. Rusu, R.B.: Semantic 3d object maps for everyday manipulation in human living environments. Ph.D. thesis, Computer Science department, Technische Universitaet Muenchen, Germany (2009)
33. Rusu, R.B., Cousins, S.: 3d is here: Point cloud library (pcl). In: IEEE International Conference on Robotics and Automation (ICRA). Shanghai, China (2011)
34. Sainz, M., Pajarola, R.: Point-based rendering techniques. Comput. Graph. **28**(6), 869–879 (2004). doi:10.1016/j.cag.2004.08.014
35. Schmid, J., Senn, M.S., Gross, M., Sumner, R.W.: Overcoat: An implicit canvas for 3d painting. In: ACM SIGGRAPH 2011 Papers, SIGGRAPH '11, pp. 28:1–28:10. ACM, New York, NY, USA (2011). doi:10.1145/1964921.1964923
36. Shakarji, C.: Least squares fitting algorithms of the nist algorithm testing system. J. Res. Natl. Inst. Stand. Technol. **103**(6), 633–641 (1998). doi:10.6028/jres.103.043
37. Shiraishi, M., Yamaguchi, Y.: An algorithm for automatic painterly rendering based on local source image approximation. In: Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering, NPAR '00, pp. 53–58. ACM, New York, NY, USA (2000). doi:10.1145/340916.340923
38. Westover, L.A.: Splatting: a parallel, feed-forward volume rendering algorithm. Ph.D. thesis, University of North Carolina at Chapel Hill (1991)
39. Xu, H., Chen, B.: Stylized rendering of 3d scanned real world environments. In: Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering, NPAR '04, pp. 25–34. ACM, New York, NY, USA (2004). doi:10.1145/987657.987662
40. Xu, H., Gossett, N., Chen, B.: Pointworks: Abstraction and rendering of sparsely scanned outdoor environments. In: A. Keller, H.W. Jensen (eds.) Eurographics Workshop on Rendering. The Eurographics Association (2004). doi:10.2312/EGWR/EGSR04/045-052
41. Yang, C.K., Yang, H.L.: Realization of seurat's pointillism via non-photorealistic rendering. Vis. Comput. **24**(5), 303–322 (2008). doi:10.1007/s00371-007-0183-y
42. Zhan, Q., Liang, Y., Xiao., Y.: Color-based segmentation of point-clouds. Laserscanning '09 **38**(Part 3/W8), 248–252 (2009)
43. Zwicker, M., Räsänen, J., Botsch, M., Dachsbacher, C., Pauly, M.: Perspective accurate splatting (2004)

Rafael Kuffner dos Anjos is a Computer Science and Engineering Ph.D. student at Técnico, Universidade de Lisboa. His doctoral studies focus mostly on Video-based Rendering. As member of the BlackBox project (sponsored by the European Research Council), he develops research on document performance composition by proposing new approaches on information representation of human performers. Other interests include 3D reconstruction techniques, data capture, and image-based collision detection. He was also a research member of the Global Lab at the National Institute of Informatics in Tokyo, being a part of the iCO2 project.

Claudia Sofia Ribeiro received her BsCS, MsC and Ph.D. in Information Systems and Computer Engineering from the Technical University of Lisbon (Instituto Superior Técnico - IST/UTL), Portugal. Currently, she is a postdoc researcher in the CLUNL Group at FCSH Lisboa, working in the ERC project "BlackBox - A collaborative platform to document performance composition: from conceptual structures in the backstage to customizable visualizations in the front-end", where she carries research and development in several topics, namely serious games, computer graphics, data capture and computer vision.

Daniel Simões Lopes has a degree in biomedical engineering from the University of Lisbon and graduated in computational engineering under the framework of the UT Austin|Portugal Program. Currently, he is a postdoc researcher and Head of Biomedical Research at the Visualization and Intelligent Multimodal Interfaces Group at INESC-ID Lisboa, where he carries research and development in several topics, namely computational geometry, contact detection, interactive computer graphics, image-based anatomical modeling, and interactive visualization of 3D medical images.



João Madeiras Pereira is Associate Professor at the Computer Science Department of the University of Lisbon (Instituto Superior Técnico - IST/UL) where he teaches Computer Graphics. João Pereira holds a Ph.D. (1996), M.Sc. and a BsEE degrees in Electrical and Computers Engineering. He coordinates the Visualization and Simulation action line of the Visualization and Intelligent Multimodal Interfaces (VIMMI) research group at INESC-ID (Computer Systems Engineering Institute). His main research fields are Real-Time Rendering, High-Performance Graphics, Serious Games and Augmented Reality. He has been involved with several European (RESOLV, IMPROVE, RTP11.13, MAXIMUS, SATIN, INTUTION, TARGET, GALA) and National projects. He is author or co-author of more than 100 peer-reviewed scientific papers presented at national and international events and journals. Professor Pereira is Senior Member of the IEEE, member of ACM and member of Eurographics Association.