Splat-based Surface Reconstruction from Defect-Laden Point Sets

Ricard Campos[a,*], Rafael Garcia[a], Pierre Alliez[b], Mariette Yvinec[b]

[a] *Computer Vision and Robotics Group, University of Girona, Spain.*
[b] *Inria Sophia Antipolis - Méditerranée, France.*

Abstract

We introduce a method for surface reconstruction from point sets that is able to cope with noise and outliers. First, a splat-based representation is computed from the point set. A robust local 3D RANSAC-based procedure is used to filter the point set for outliers, then a local jet surface – a low-degree surface approximation – is fitted to the inliers. Second, we extract the reconstructed surface in the form of a surface triangle mesh through Delaunay refinement. The Delaunay refinement meshing approach requires computing intersections between line segment queries and the surface to be meshed. In the present case, intersection queries are solved from the set of splats through a 1D RANSAC procedure.

Keywords: Surface reconstruction, splat-based representation, RANSAC fitting, Delaunay refinement.

1. Introduction

The growing variety of scanning devices and technologies nowadays provides measurements of objects in the form of point sets. Despite the advances in scanning technologies, achieving a perfect scan is very unlikely and, in general, these point sets contain two main types of defects: noise and outliers. Noise refers to the inaccuracy of the resulting measurements, and is related to the lack of precision and repeatability of the scanning process used, while outliers are wrong measurements that are produced by errors during the scanning (e.g., reflections on the surface when using laser scanners).

Having the object represented as a point set complicates the interpretation of the data since the notion of visibility of the shape from a given viewpoint is lost. For this reason, surface reconstruction methods are required. We aim at obtaining the surface in the form of a surface triangle mesh, which allows further processing such as remeshing or simplification. In this paper, we present a surface reconstruction method that is able to provide a smooth approximation of the sampled shape while dealing with noise and a large number of outliers. Moreover, the method is able, to some extent, to reconstruct surfaces with boundaries.

2. Related work

Surface reconstruction methods are broadly divided according to their ability to interpolate or approximate the measured points.

Interpolation-based methods have been proposed mainly by the computational geometry community. They commonly rely on extracting the surface from a space partitioning such as the Delaunay triangulation and its dual, the Voronoi

*Corresponding author.
Email addresses: rcampos@eia.udg.edu (Ricard Campos), rafael.garcia@udg.edu (Rafael Garcia), pierre.alliez@inria.fr (Pierre Alliez), mariette.yvinec@inria.fr (Mariette Yvinec)

diagram. Well-known methods in this area include the Cocone method [1], and the Power Crust method [2]. They use a principled way to separate the tetrahedra that are part of the inside volume of the object from those that are part of the outside volume, and then extract the surface as the triangles that define the interface between these two volumes. Other methods, such as ball-pivoting [3], generate the surface incrementally, starting from a single triangle and adding new triangles to the surface by making local decisions. The main limitation of interpolation-based methods is that they are not able to deal with noise. Since the input points (or a subset of them) become the vertices of the reconstructed surface, the quality of the surface reflects the noise in the input points. This requires preprocessing to smooth out the noise before resorting to an interpolative method. The method introduced by Digne et al. [4] improves the results of the ball-pivoting method by applying it to a scale-space version of the point set. Also, some interpolation-based methods are effective against outliers. The method introduced by Kolluri et al. [5], based on spectral graph partitioning on a Delaunay triangulation, achieves excellent robustness when faced with a large quantity of outliers (up to 5%). Graph partitioning techniques have also been used by Labatut et al. [6], and achieve further resilience to outliers by relying on additional information such as sensor locations and lines of sight.

Regarding approximation methods, a great majority of them define the surface in implicit form before evaluating it at each position inside a regular voxel grid. Then, through an isosurface extractor method that is usually a variant of marching cubes [7], the surface is recovered as a surface triangle mesh. Early contributions in this area were presented by Hoppe et al. [8], where a local process is used to compute normals before orienting them through greedy propagation. From all input points with their oriented normals and local tangent planes, a signed distance function is computed inside the voxel grid by computing the signed distance from each voxel to its nearest tangent plane. Another proposal by Curless and Levoy [9] uses the notion of local connectivity present in a single range scan to perform a space carving technique. Other methods are based on surface deformation [10], where a first rough approximation of the signed distance field representing the surface is evolved towards a minimum representing the inferred shape. The method proposed by Hornung and Kobbelt [11] uses a graph partitioning method to extract the surface from an unsigned distance function.

Other types of approximation methods use Radial Basis Functions (RBF) to approximate the signed distance function from the input points [12]. These RBF can be fully supported [12] or compactly supported [13], but in both cases, per-point normals are required.

Kazhdan et al. proposed a method [14] which looks for an indicator function, being zero outside the inferred model and one inside. They consider input points and normals as samples of the gradient of the indicator function, and cast the problem as finding the solution of the Poisson equation. Additionally, it uses an octree to lower memory consumption compared with a regular grid.

An increasingly popular approach is the use of Moving Least Squares (MLS) surfaces. The initial definition as an interpolation technique from unstructured data was proposed by Levin [15]. Alexa et al. [16] used this approach for direct visualization of surfaces defined by point sets. Most of these MLS surfaces have an implicit form that can be triangulated by an isosurface extractor method. Other approaches achieve better resilience to noise, as with the Algebraic Point Set Surfaces, introduced by Guennebaud and Gross [17]. Inspired by satisfactory results in recovering smooth surfaces, some efforts in recent years have been directed towards piecewise-smooth surfaces, i.e., preserving sharp features into the MLS definition [18].

While most approximation-based approaches provide an implicit noise correction to some extent, they rarely cope with outliers. Because of their global nature, methods such as Poisson reconstruction achieve resilience to small quantities of outliers, but fail with a large number of outliers. Only a few methods exhibit robustness to

both noise and outliers. Mullen et al [19] use an outlier-robust unsigned distance function that is then signed through a sequence of heuristics. In order to achieve high robustness to outliers, there is an increasing use of robust statistical methods such as the popular RANSAC [20]. In the surface reconstruction literature, RANSAC is often used for shape detection when the searched object is composed of canonical shapes such as planes or spheres, and extracts them sequentially [21].

It should be noted that an important requirement of many surface reconstruction methods is knowing the normals at each input point. Since not all scanning systems provide normals, they must be computed prior to using one of these surface reconstruction methods. However, computing the normals is also an ill-posed problem, since estimating the normals requires inferring the surface locally. Following the most common approach presented by Hoppe et al. [8], many methods aim at computing reliable normals in the presence of noise. Mitra et al. [22] use an adaptive k-sampling around the data, and provide theoretical bounds on the best radial neighborhood to select. Another popular approach for normal estimation consists of using Voronoi poles, as proposed by Amenta and Bern [23]. Under ideal sampling conditions, the Voronoi cells generated from the input points are elongated in the normal direction. Thus, the vector formed by the point and the Voronoi vertex farthest from the point (the pole) is a good approximation of the normal vector. However, noise in the data prevents this from happening. In order to deal with noisy data, Dey and Sun [24] propose using the nearest large Delaunay balls to compute the normals. By mixing both previous proposals, Alliez et al. [25] compute the normals using PCA from the space defined by a union of Voronoi cells near the input points. Finally, regarding robust statistics, the method presented by Li et al. [26] relies on noise-scale estimation to obtain a robust local tangent plane estimation which requires less parameter tuning than the common RANSAC procedure. It is worth noticing that all these methods further require another heuristic to achieve a globally consistent orientation for all the normals.

Finally some conclusions can be drawn from the state-of-the-art review presented. We have seen that the great majority of approximation-based methods rely on oriented normals to better deal with noise, but they are not always available when using scanning technologies and are difficult to estimate from raw point sets. We also observe that the literature is considerably thinner on the question of robustness to outliers. We also note that while some interpolation-based methods are able to deal with outliers to some extent, they are not able to deal with noise. Thus, devising methods that are simultaneously resilient to both noise and outliers and that do not depend on a priori knowledge (such as normals) is direly needed. Our end goal is to devise more general methods applicable to inputs coming from a wider variety of measurement devices and processes.

3. Overview and Contributions

Our method is based on computing a global surface approximation using local surfaces. Instead of producing a consistent global representation of the surface through a memory-intensive global solver (such as solving for a signed distance or indicator function) and then using an isosurface meshing approach, our approach performs the merging of the different local surfaces at the meshing step.

We denote as *splats* these local surfaces, which may not be just planar but higher-degree approximations instead. A splat, computed using a local neighborhood of the input points, takes into account the fact that the input point set may be corrupted with noise and outliers. From the local neighborhood of an input point, we use the RANSAC method to determine which of the points are most likely to be part of the surface. The splat is then computed using least-squares fitting of a differential jet. Thus, the differential jet approximation handles the effects of noise in the data, while RANSAC ensures using only inlier data in their computation.

The set of splats provides a global approximation of the sampled shape, amenable to coarse-to-fine meshing through Delaunay refinement [27].

The main advantage of the Delaunay refinement method is that it only requires computing intersections between line segment queries and the surface to be meshed. In our case, the surface-segment intersection queries are robustly solved by computing splats-segment intersections and running another 1D RANSAC procedure along the segment query. Even if small inconsistencies are present in the splats representation, RANSAC takes advantage of the redundancy between splats so as to be able to answer correctly the intersection query required by the surface meshing method. Figure 1 depicts an overview of our method.

It is worth noting that coherent orientation between splats is not required, since only intersection points are considered during meshing regardless of their orientation. Thus, the method works on raw pointsets, avoiding the requirement of per-point normals from most of the approximation-based methods of the state of the art. This makes the method amenable to point sets coming from a wider range of sources.

In addition to a low memory footprint, our two-step proposal allows us to generate the final mesh at different resolutions given the same splat-based representation by only changing the parameters of the meshing algorithm. Besides, the splat-based procedure allows the user to select the fitting degree of the splat as a means to trade smoothness for fitting accuracy to the input data.

Finally, because of the limited support of the splats, the method does not find a surface over areas that are not sampled, allowing the recovery of surfaces with boundaries. Although hole filling capabilities are desirable in some cases, guessing the surface on large undersampled areas often leads to artifacts on the final surface.

This paper is structured as follows: Section 4 details the creation of the splat-based representation from the input point set. Section 5 describes the surface meshing step, through robust merging of the splats. Section 6 illustrates the robustness and versatility of our method. Section 7 presents some conclusions and future work.

4. Creating the Splats

Given the input point set P, we compute a splat-based representation in which each splat is a local approximation of the surface. In its simplest form, a splat is a disk tangent to the surface and with a radius adapted to the local density of the point set. However, our method allows higher-degree approximations through so-called *jet surfaces*. We next explain how these jet surfaces are computed and how we combine them with RANSAC in order to achieve robustness to outliers.

4.1. Local Jet Surfaces

For each point $p_i \in P$, we pick its k nearest neighbors $K(p_i)$. In each local neighborhood we then compute a local smooth jet surface as introduced by Cazals and Pouget [28]. A jet surface is defined as a least squares approximation of a height function in a local reference frame. Coherence between neighboring splats is achieved through overlapping neighborhoods, and robustness to noise is achieved through least squares approximation.

More specifically, given a subsample of the points $K(p_i)$, and a local coordinate framework, the jet surface is defined as the Taylor expansion of a height function (truncated to a given degree). It is represented as:

$$z(x, y) = J_{B,d} + h.o.t., \tag{1}$$

where $h.o.t.$ stands for higher order terms, and $J_{B,d}$ is a jet surface of degree d (a d-jet) corresponding to:

$$J_{B,d} = \sum_{k=0}^{k=d} \left(\sum_{i=0}^{i=k} \frac{B_{k-i,i} x^{k-i} y^i}{i!(k-i)!} \right). \tag{2}$$

Such a local representation of a smooth surface is valid as long as the z-axis of the local coordinate framework is not included in the surface tangent plane. It may be computed from $K(p_i)$ by least square fitting. To get an accurate estimate of the jet surface at p_i we use a local coordinate framework obtained from $K(p_i)$ by principal component analysis (see Figure 3(a)).

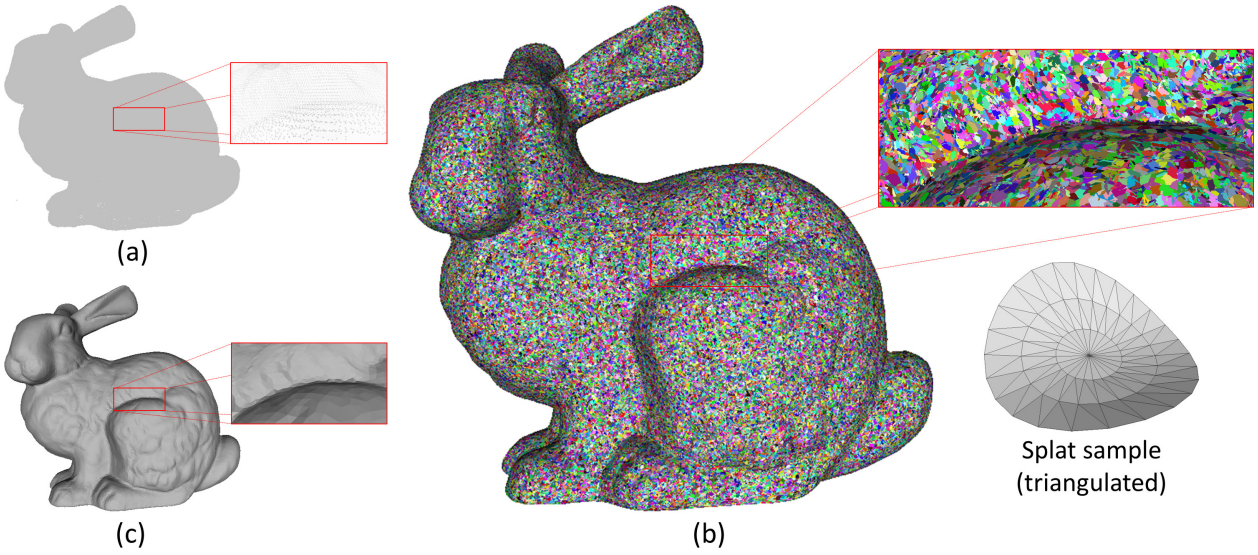


Figure 1: Overview of our method. (a) Input raw point set. (b) Splat representation (discretized using triangle fans for visual depiction). (c) Output surface triangle mesh generated through Delaunay refinement.

From the jet surface we extract another representation, referred to as the *Monge form*. The Monge form is the Taylor expansion of the height function in the coordinate system whose axis are aligned aligned with the normal and the principal curvature directions of the surface.

$$\begin{aligned} z(x,y) \quad = \quad & \tfrac{1}{2}(k_1 x^2 + k_2 y^2) \\ & + \tfrac{1}{6}(b_0 x^3 + 3b_1 x^2 y \\ & + 3b_2 x y^2 + b_3 y^3) \\ & + \tfrac{1}{24}(c_0 x^4 + 4c_1 x^3 y + 6c_2 x^2 y^2 \\ & + 4c_3 x y^3 + c_4 y^4) + h.o.t. \end{aligned} \tag{3}$$

In the original method [29], this Monge form is used to evaluate differential properties of the surface at a query point, such as principal curvatures and directional derivatives. In our framework, we use it because it describes the local surface using fewer terms and will reduce the cost of computing intersections with segments during the Delaunay refinement meshing phase. Observe indeed that for a jet surface of degree $d = 2$, the number of coefficients in the jet surface is a priori $c = (d+1)(d+2)/2 = 6$, while the Monge form involves only two coefficients.

We now comment on the difference between our approach and the MLS surfaces, since both approaches involve local computations around a query point. The second part of the projection operator presented by Alexa et al. [30] is also a local bivariate surface computation. However, the jet surface definition is explicit: it defines a fixed surface around a given query point, while MLS projection operators vary depending on the query point, which requires an iterative procedure for each intersection query with line segments. In Section 5 we explain how our framework takes advantage of the fact that the intersection query between a segment and a local surface of degree up to 2 can be computed more directly.

4.2. Outlier Rejection

For outlier rejection we use an approach based on 3D RANdom SAmple Consensus (RANSAC) [20]. The RANSAC method computes models from data corrupted with large quantities of outliers. At each iteration, it instantiates a model using a random sample of s points, where s is the minimum number of points needed to compute the model. All remaining points are tested against the current model, i.e., they are considered to be compatible with the model when they are within a maximum tolerance error from the model. Each point agreeing with the model votes so as to consolidate a consensus defined by the number of points agreeing with the model. After repeating the process a number of iterations, the model having largest consensus (support) is selected and a refined least-squares solution for the

model is computed from all agreeing data. The intuition is that models generated from points that include outlier data should have little (weak) support while those generated from inlier data should have greater support.

In our algorithm, the jet surface computation is used as the model that RANSAC fits to $K(p_i)$. At each iteration inside the RANSAC procedure, a minimum set of points from $K(p_i)$ is selected and a model is computed from that. Then, the votes for this model are the number of points closer than δ_r from it. Note that since the surface can be of an arbitrary degree, we use the algebraic distance instead of the common Euclidean one, provided that it is more efficient to compute. Furthermore, a minimum number of inliers (δ_m) is required to accept the computed jet, otherwise the model is too weakly supported, and thus the points defining it should be considered as outliers. At the end of the RANSAC method, if a point p_i does not fit the best local jet (with the largest support), or the computed jet is too weakly supported, it is considered as an outlier and no further processing is carried out.

We will use a probabilistic approach to determine the number of iterations RANSAC needs to generate a model. As it is computationally too expensive to try each possible set of samples to build a model, we instead aim at finding the minimum number of iterations (N_{iter}) that ensures that, with probability q, RANSAC will pick at least one sample of size s free from outliers. If we define w as the probability that any point inside the currently selected sample is an inlier, the probability of this point being an outlier is $\epsilon = 1 - w$ [31]. Thus, the number of iterations is bounded by:

$$N_{iter} = \log(1 - q)/\log(1 - (1 - \epsilon)^s). \qquad (4)$$

We choose to fix $q = 0.99$ in our experiments. Since we do not know beforehand the percentage of outliers inside a given $K(p_i)$, we initialize ϵ with a worst-case estimate, which has been fixed to $\epsilon = 0.5$ for all experiments shown. Then, at each iteration, we update this estimate if the current computed model has greater support. The minimum number of points required to generate the jet surface depends on the degree of the jet. The number of coefficients c of a jet surface, and consequently the minimum number of points for the fit, is defined by $c = (d + 1)(d + 2)/2$.

From the $K(p_i)$ set of points, the RANSAC procedure extracts its subset of inliers, which are referred to as $I(p_i)$. In a final step, RANSAC computes the final jet-surface using all the points in $I(p_i)$.

4.3. Splat Sizing

Although each of the splats has finite support, we favor redundancy between neighboring splats since we take advantage of this property during meshing. The size of the splat is simply computed from $I(p_i)$ as the mean distance from each sample $p_j \in I(p_i)$ to p_i so that the size of the splat depends on the local point set density.

5. Meshing

We have generated one splat for each input point where RANSAC was successful, and because of the redundancy between them, the same area may be covered by more than one splat. In order to obtain the final surface triangle mesh from the splats, we use a coarse-to-fine meshing algorithm based on the concept of Restricted Delaunay Triangulation (RDT) and Delaunay refinement [27]. Given a set of points E on or near a surface, the RDT is a subcomplex of the 3D Delaunay triangulation of E formed by the Delaunay triangles whose dual Voronoi edges intersect the surface. Each RDT triangle has a circumball centered on the surface and empty of all other E points. This circumball is called a surface Delaunay ball (see Figure 2).

Boissonat and Oudot [27] proved that if the sampling E of the surface is dense enough with respect to the local feature size of the surface, the RDT provides a good approximation of the Hausdorff distance to the surface. Moreover, it provides a good approximation of normals, areas and curvatures. The meshing algorithm iteratively refines an initial 3D Delaunay triangulation

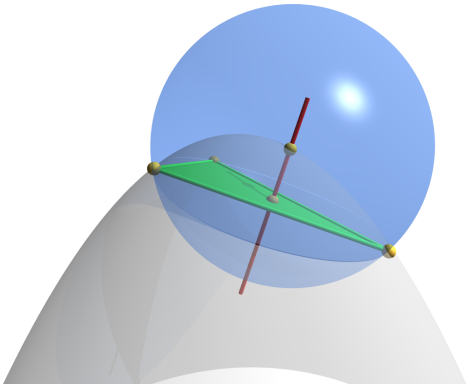


Figure 2: Example of a surface Delaunay ball.

until all surface Delaunay balls meet some properties. More specifically, starting from a small set of points on the surface, the method inserts the center of a *bad* surface Delaunay ball at each iteration. A surface Delaunay ball is considered bad when it does not meet any of the following requirements:

- Angle bound (α_a): The RDT Delaunay triangle inside the ball must have all its angles larger than this angle bound.
- Radius bound (α_r): The ball must have a radius lower than this bound.
- Distance bound (α_d): The distance between the center of the ball and the circumcenter of the associated RDT triangle must be lower than this bound.

The algorithm terminates when the RDT contains no triangle with a bad surface Delaunay ball. The RDT is then the targeted approximation of the surface, and tuning the criteria defining bad Delaunay balls is our means to control the quality of the approximation and of the output isotropic surface triangle mesh in terms of sizing and shape of the triangles.

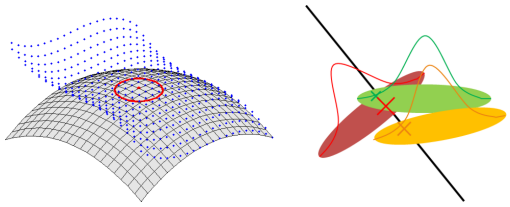
In our context, an interesting property of this meshing approach is that it only requires devising an oracle that, given a Voronoi edge, i.e., a line segment query, computes its intersection with the surface. This loose requirement makes the algorithm well-suited to various application scenarios. For example, Salman and Yvinec [32] used this surface meshing approach to recover a surface directly from an unstructured triangle soup. Note also that its coarse-to-fine procedure maintains a low memory consumption.

Calculating the intersection between a line segment query and a jet surface of $d > 1$ has a high computational cost. For this reason, we use a planar disc version of the splat as the first approximation for efficient intersection detection. This planar approximation is defined by the origin of the splat and the normal at this origin (both defined by the computed Monge form of the jet surface), and the radius computed in section 4.3. Prior to meshing, we insert all the disks into a hierarchical AABB tree data structure (Axis-Aligned Bounding Boxes) in order to further accelerate the intersection detection. Once we detect the discs intersected by a segment query, we proceed as follows for each disc. We move the intersection points along the segment query in order to refine the intersection points with the associated local higher degree jet surfaces. Note that when the local surface has degree $d = 2$, there is a closed form solution [33], and local surfaces of degree $d > 2$ require solving a non-linear minimization along the segment. Starting from the intersection point with the disc approximation, the intersection point is moved towards the first intersection point along the segment using Levenberg-Marquardt optimization. We observe that higher order local surfaces are useful in undersampled datasets in which local changes in the surface are more abrupt and thus require more complex local surfaces. However, we also observe that using local surfaces of high degree produces surfaces that capture the noise instead of correcting it.

For initialization an initial subset of the centers of the computed splats, 20 in all results shown, are used to create the initial triangulation required to seed the mesh refinement process.

5.1. Merging Local Intersections

If the input point set is dense enough, it is very likely that each intersection query returns not just one but a set of intersection points. Assuming for the moment that there are no outliers among these intersection points, we need to provide a sin-



(a) (b)

Figure 3: Intersecting a splat with a line segment. (a) A jet surface (in gray) is fitted to the neighborhood of the red point. The limit of the splat support is depicted with a red circle. Blue points depict the input points. Observe how the approximation of the splat is faithful closer to its center. (b) Several disks are intersected by the query segment. The contribution of each intersection point is weighted by a function of its distance from the center of the splat. The lines over the discs depict the Gaussian weights used.

gle intersection point summarizing the local splats intersected. We first observe that the local surface computation is "variability centered", i.e., more reliable when close to the center of the splat. We compute a weighted mean to reflect this observation (Figure 3(b)).

Each splat is assigned a Gaussian weight function located at its center. Each intersection point is then given a weight according to its distance from the center of the splat:

$$w(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \tag{5}$$

where $\sigma = r_i\gamma_g$, and r_i denotes the radius of the splat i (its size) and γ_g denotes a user parameter used to adjust the scale of the Gaussian. The final merged intersection point p_{int} is computed as:

$$p_{int} = \frac{\sum_{i\in Q} w(\|p_i - o_i\|)p_i}{\sum_{i\in Q} w(\|p_i - o_i\|)}, \tag{6}$$

where Q denotes the set of intersected splats, and p_i and o_i denote the intersection point and the origin of the intersected splat, respectively.

5.2. Robust Intersection Query

Despite our efforts for achieving coherence between neighboring local splats, the weighted averaging process described above is not appropriate for high levels of noise. In addition, the local approximations of the surface are computed from a given center (an input point), thus their accuracy decreases with their distance from this center. Furthermore, depending on parameter δ_r, the RANSAC procedure described above is not able to detect outliers that are far from the true inlier points yet close to the jet surface approximation. For these reasons, we need an intersection test robust to both low-conformity among neighboring splats and remaining outliers.

We have one splat for each input point, hence, due to splat overlaps, the surface is locally represented by more than a single splat. Intuition tells us that, in general, most of the splats are local faithful approximations of the surface, except for a few outliers. Figure 4(a) depicts this general configuration: most intersection points are within one cluster, very close to one another, while outlier intersection points are isolated. We thus aim at extracting the final intersection point by taking into account the cluster of points with the largest support.

In order to extract this cluster, a 1D RANSAC procedure is applied along the intersection query where the model is a weighted centroid. At each iteration of the RANSAC procedure, two points are randomly selected, the minimum to define a centroid, and a model is instantiated from them. Then, we rank this model according to the number of points being closer to the centroid than a distance parameter δ_i. Two sample iterations of the algorithm are illustrated in Figures 4(b,c). At the end, the centroid having largest support is selected, providing the cluster of points used to compute the weighted averaging as described above. A percentage of the length of the segment query is used to define the δ_i parameter.

Note that this procedure requires at least two splats overlapping a given area. If a single splat is intersected, we return no intersection as no consensus can be established locally. Redundancy among splats is thus central to provide robustness in our approach.

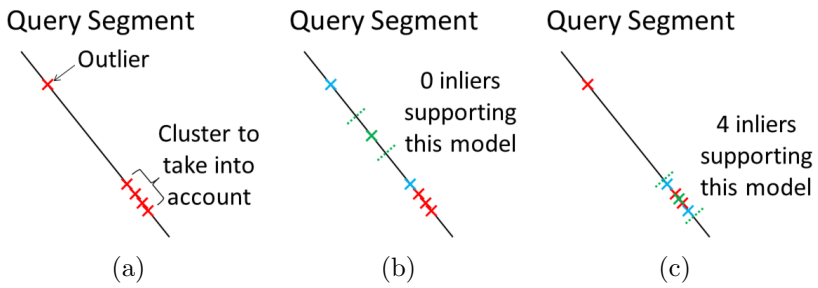


Figure 4: Robust intersection. (a) The splats that are coherent among each other generate a cluster of intersection points, while a non-coherent splat generates an isolated intersection point, classified as an outlier. (b) and (c) illustrate two possible iterations of the RANSAC algorithm, centroids (green) generated from outliers have weaker support than the ones generated from inliers.

6. Results

We implemented our method using components from the CGAL library [34]. Our current implementation is sequential. Table 1 lists all the parameters used by the two steps of our algorithm. Note that some parameters are not critical and thus have been set once for all experiments shown. The data-dependent parameters are k (nearest neighbors) and those driving the splat-RANSAC procedure. There are 3 additional parameters required for meshing. All the parameters used to generate the results are listed in Table 3, along with the running times for the two steps of the algorithm. Note that we choose similar values for α_r and α_d, because this leads to surfaces with triangles of similar size.

As illustrated in Figure 5, our method works well in the ideal scenario, with no noise or outliers. The Stanford bunny dataset is chosen to illustrate the behavior of our method under harder noise conditions (Figure 6). The Figure 7(a) confirms that our method is already able to deal with noise with local surfaces of degree 1. The other subfigures show the results obtained when increasing the degree. The timings listed in Table 3 confirm a rapid decrease in performance for degrees higher than 2, without visible increase in the quality of the results. In the remainder we report results only for local surfaces of degree $d = 2$.

Robustness to unstructured outliers is illustrated in Figure 8. We artificially add outliers to the Stanford bunny point set by adding uniform random points inside a loose bounding box (enlarged by 5% of its diagonal) of the original point set. The second row shows the results when applying our method with a less restrictive δ_r. In the presence of outliers, this parameter becomes relevant as it is the one defining the inlier set at each RANSAC iteration. Using a looser threshold leads to outlier points close to the surface, which generate outlier splats. These outlier splats produce holes in the final surfaces since they are not coherent with the other splats, and they are not detected for intersection by our robust intersection query. The third row in Figure 8, where δ_r is set to be more restrictive, shows that the results are quite similar for all outlier-ridden datasets. When taking a look at the running times of these datasets in Table 3, we note that there is a noticeable decrease in performance during the creation of the splats. This is due to the RANSAC procedure being unable to obtain a result with enough inliers to estimate the number of iterations ruled by Equation 4 to decrease. This means that RANSAC reaches the maximum number of iterations to decide finally that there is no valid jet surface possible at a given outlier point.

In order to measure the accuracy of our method for variable amounts of noise and outliers, a synthetic dataset consisting of a unit sphere has been generated. Table 2 and Figure 9 show the results for added Gaussian noise and outliers. Table 2 shows that noise makes the accuracy of the final surface decrease incrementally and adds non-manifoldness caused by the non-conformity between neighboring splats. For a high level of noise, the method is not able to recover a good splat approximation and starts degrading. However, even

Parameters		
Splats		
Name	Description	Default Value
k	Number of neighbors to take into account	(variable)
d	Degree of the splat fit	2
δ_r	RANSAC distance to plane threshold	$0.01BBD$
δ_m	RANSAC minimum number of inliers	(variable)
Meshing		
Name	Description	Default Value
α_a	Surface meshing angle bound	10
α_r	Surface meshing radius bound	(variable)
α_d	Surface meshing distance bound	(variable)
δ_i	RANSAC threshold for the intersection query	$0.05QSL$
γ_g	Gaussian deviation factor	0.25

Table 1: Parameters used by the two steps of our algorithm. We also specify the constant values used in all the experiments shown in Section 6. BBD denotes the Bounding Box Diagonal and QSL denotes the Query Segment Length.

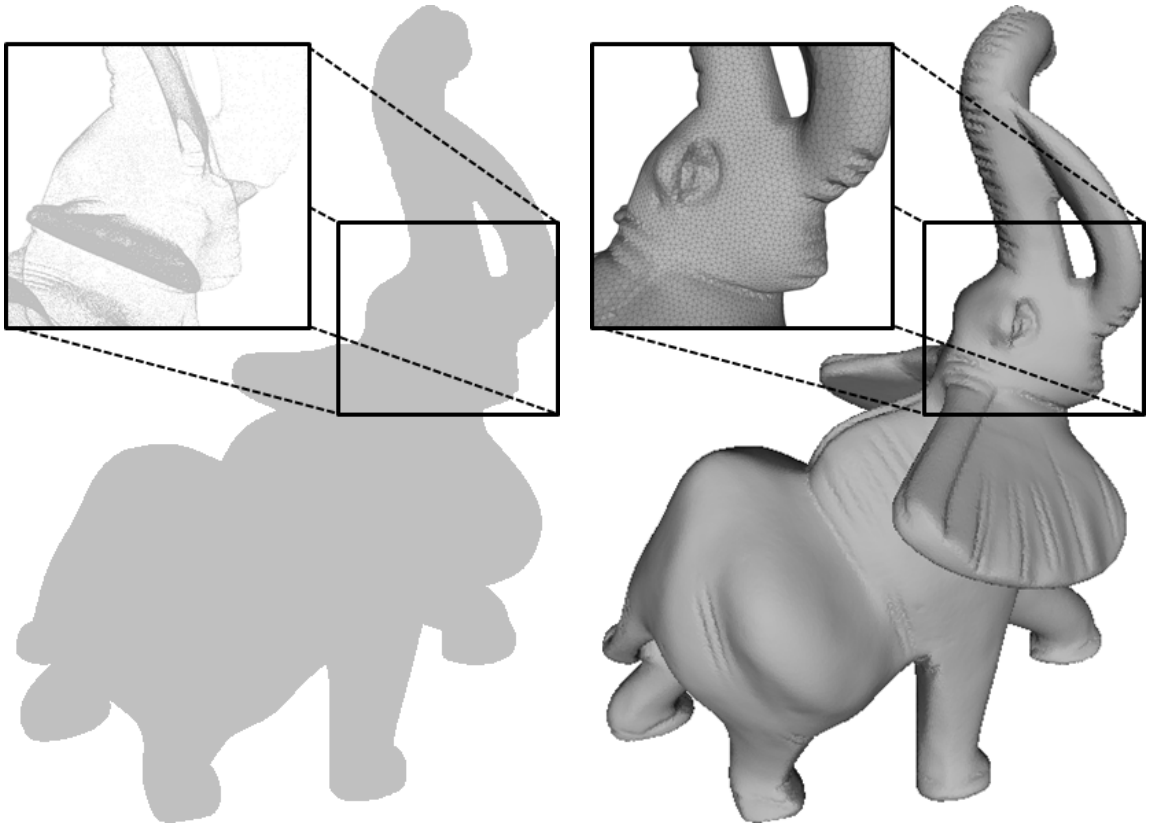


Figure 5: Elephant dataset: 1,537,283 points, both noise and outlier free. Left: input pointset with closeup. Right: reconstructed surface with closeup.

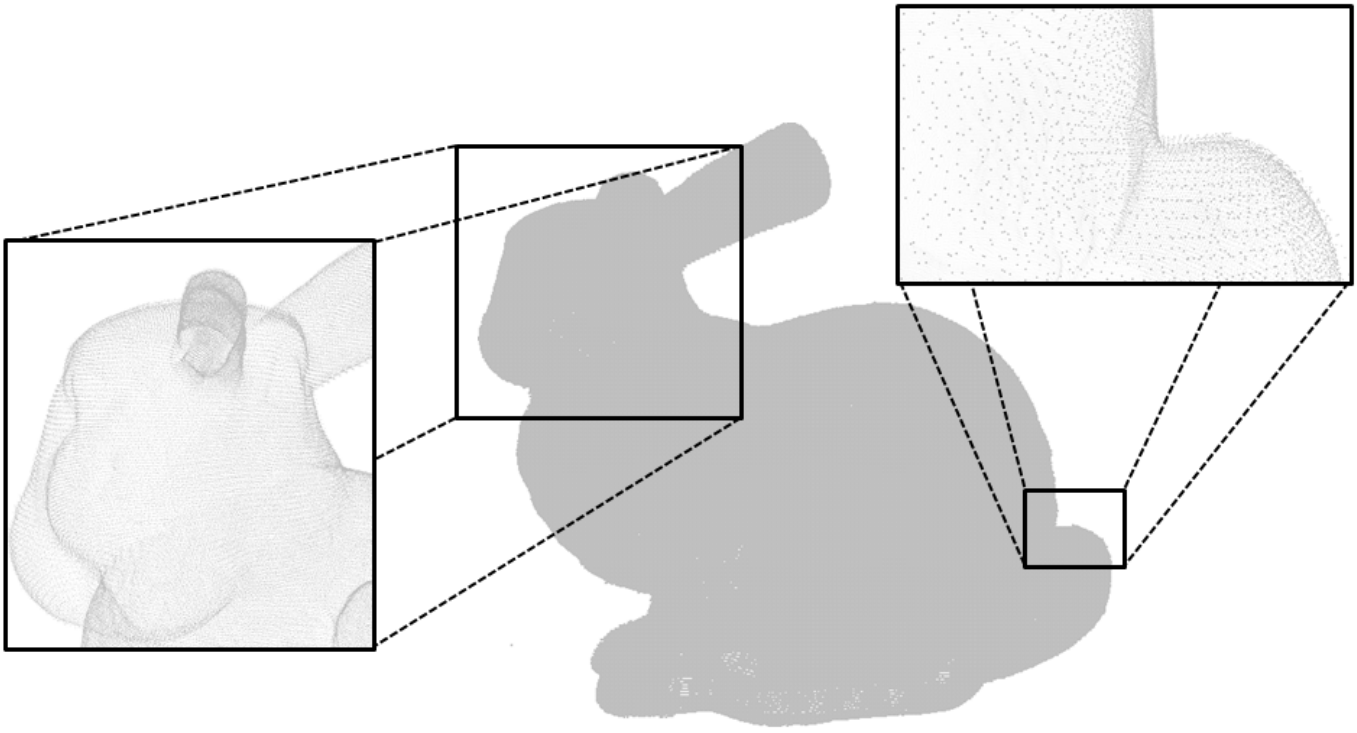
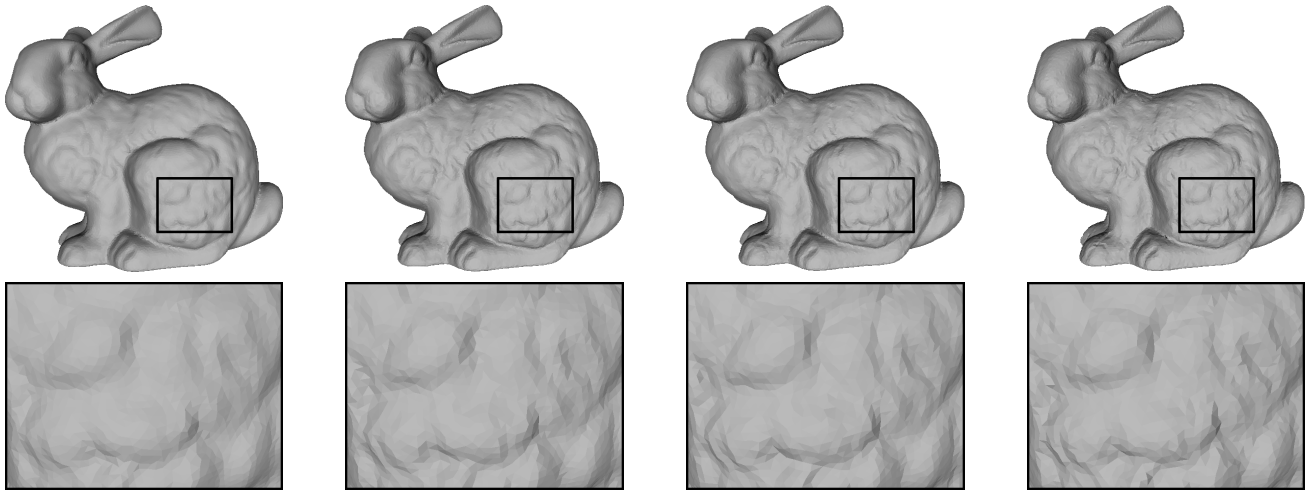


Figure 6: Stanford bunny: 362,272 points with a high level of noise.



(a) $d = 1$ (b) $d = 2$ (c) $d = 3$ (d) $d = 4$

Figure 7: Stanford bunny. Reconstructed surfaces with a gradual increase of the degree of the splats.

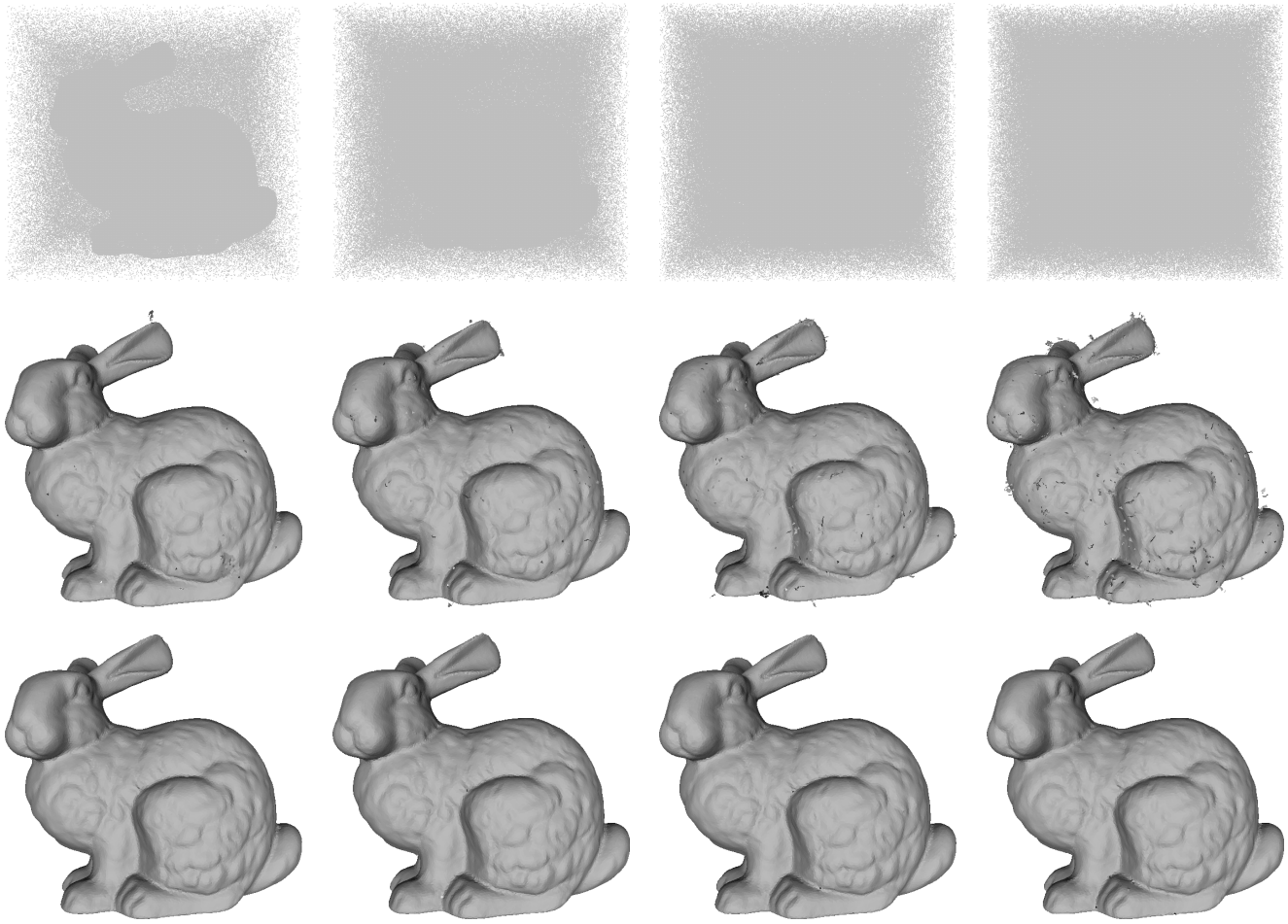
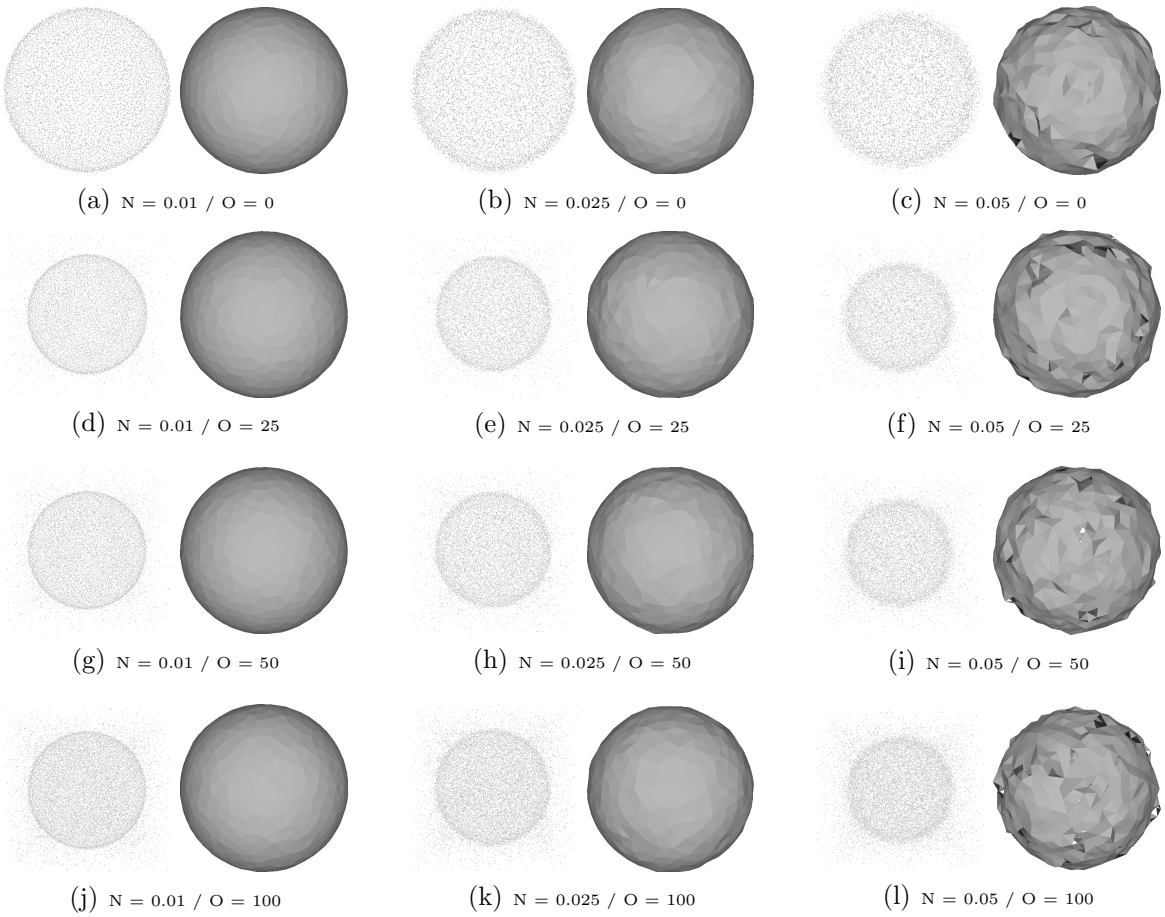


Figure 8: Robustness to outliers. The first row shows the outlier-ridden point sets. The percentages of outliers added, with respect to the number of points on the original point set, are 50%, 100%, 150% and 200%, added inside its bounding box (enlarged by 5%). The last 2 rows show the results when applying our method to its vertically corresponding point set. The second row uses $\delta_r = 0.005BBD$, while the third row uses a more restrictive $\delta_r = 0.0025BBD$.

Noise	Outliers	Vertices	Edges	Mean Error	Min Error	Max Error	NM Edges	NM Vertices
0	0	840	1676	2.33e-05	1.86e-05	4.16e-05	0	0
0.01	0	829	1654	0.001438	1.86e-06	0.005201	0	0
0.01	25	818	1632	0.001620	3.80e-07	0.006418	0	0
0.01	50	837	1670	0.001926	8.77e-06	0.007822	0	0
0.01	100	824	1646	0.002120	3.25e-06	0.010432	4	0
0.025	0	830	1664	0.004195	6.16e-06	0.016708	16	0
0.025	25	830	1662	0.004322	1.10e-05	0.022721	12	0
0.025	50	821	1642	0.004567	2.17e-06	0.023205	8	0
0.025	100	829	1666	0.004980	1.27e-05	0.023553	23	0
0.05	0	864	1749	0.013898	5.82e-05	0.063856	117	8
0.05	25	868	1752	0.013898	1.51e-06	0.093498	123	11
0.05	50	868	1769	0.013716	1.93e-05	0.074861	162	11
0.05	100	864	1708	0.015326	1.04e-05	0.090198	133	16

Table 2: Robustness on synthetic sphere point set. The 1st column shows the added noise (standard deviation), the 2nd, the percentage of outliers added according to the original point set. The 3rd and 4th columns indicate the vertices and edges of the resulting surface. The columns from the 5th to the 7th show the mean/min/max error of the points on the surface from the original unit sphere, while the last two columns correspond to the number of non-manifold edges and vertices in the resulting surface.



(a) N = 0.01 / O = 0 (b) N = 0.025 / O = 0 (c) N = 0.05 / O = 0

(d) N = 0.01 / O = 25 (e) N = 0.025 / O = 25 (f) N = 0.05 / O = 25

(g) N = 0.01 / O = 50 (h) N = 0.025 / O = 50 (i) N = 0.05 / O = 50

(j) N = 0.01 / O = 100 (k) N = 0.025 / O = 100 (l) N = 0.05 / O = 100

Figure 9: Synthetic unit sphere dataset. Results for increasing levels of noise and outliers. The original point set was made out of $10,242$ points. N stands for noise (standard deviation) and O stands for outliers (percentage). These results correspond to the data in Table 2.

in this case the effect of outliers is not noticeable.

Despite a high resilience to noise and outliers, and given the local nature of the primitives used to approximate the inferred surface locally, our method is not always able to fill in holes in the resulting surface. This is bad in some cases where it would be desirable to fill small holes in the surface due to missing data, but it is also useful to get no reconstruction in some parts when the goal is to reconstruct surfaces with boundaries. The foot dataset depicted in Figure 10 illustrates this dilemma.

Our mesh generation procedure provides an easy way to obtain different resolutions for the final surface mesh by changing the Delaunay refinement parameters (see Figure 11). Methods relying on variants of the marching cubes commonly require re-creating the voxel grid at the desired resolution in order to modify the properties of the output mesh, while our splat-based representation is unchanged.

So far, the method has been tested against regularly and densely sampled datasets. Figure 12 shows the results of the method for a sparse dataset. After applying the method with $k = 25$, the resulting surface contains some holes; they can be seen in Figure 12(b). This is due to undersampled areas. Observe that it makes no sense to apply the RANSAC method presented in Section 5.2 when there is only one intersected splat, since it is impossible to tell whether this intersection point is an outlier or not. Thus, at least two splats are required to tell if an area is correctly represented (intersection points are close to one another), and three or more are needed to run the RANSAC procedure. One may think that increasing k would also make the resulting splats bigger and cover the holes. This is what happens in Figure 12(c), when we use $k = 50$, although there are still some holes left. Note however that increasing k too much, as depicted by Figure 12(d), leads to bad local fitting and hence bad fitting between neighboring splats. As a consequence, more holes are generated because of our RANSAC intersection test. This issue, left for future work, suggests that a fixed k may not be a correct solution for non-uniformly sampled datasets, and should be adaptive to sampling density.

Figure 13 shows how some of the state-of-the-art methods behave when applied to the Stanford bunny dataset. They can be compared with the ones presented in Figure 7. We observe that the interpolatory methods, Robust Cocone [35] and Power Crust [2], are not able to deal with noise. Regarding the methods based on approximating the surface, they all achieve smoother results. The results obtained by APSS [17], Poisson [14], VRIP [9] and MPU [36] are comparable, but MPU returns an over smoothed surface in some parts. Regarding the APSS method, it depends on the scale parameter. If this scale parameter is too low, it generates a fitting too tight to the input points and causes some noise to be captured in the surface. Our method recovers a surface very similar to that obtained by Poisson and VRIP, but does not require any additional information. Poisson, MPU and APSS require oriented normals at input points, while VRIP requires the knowledge of the individual scans forming the point set.

Finally, to test our method to its limit in terms of robustness, we run it on point sets obtained from images (pedestrian and aerial) through dense computer vision photogrammetry methods. The input point sets contain a large number of outliers, since they were generated using a naive dense image matching technique applied to a large number of pairs, where the outliers add up for each pair. Furthermore, many of them are structured outliers (e.g., they are located on the same plane), and not just uniformly spread as assumed previously. Figure 14, shows results obtained on a column capital. Despite the structured outliers present in the input data, our method is able to remove the vast majority of them, generating a few wrong splats in the areas near the inferred surface, which then translate into wrong off-surfaces. Figure 15 depicts a large scale aerial measurement of the abbey of Cluny (France). The point set presents even more difficulties, since in addition to the large number of structured outliers, the sampling is both widely variable and anisotropic. These differences in sampling are due to the variable distance from the building to the camera: buildings closer to the

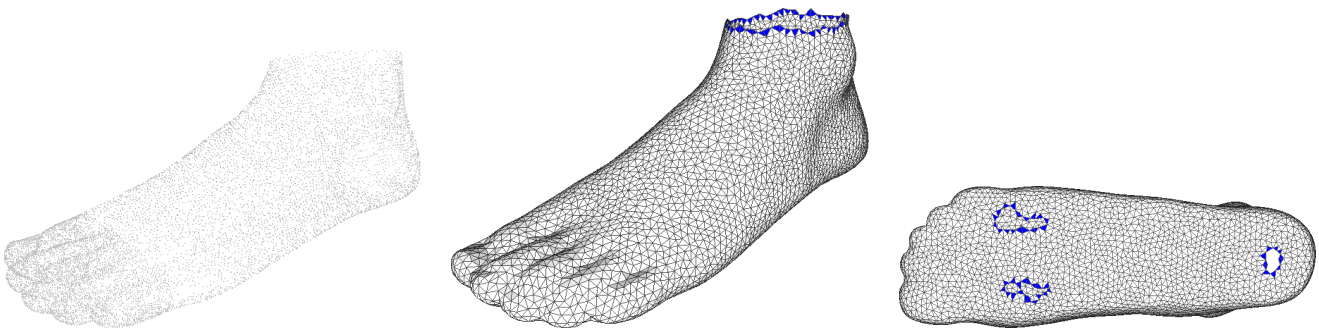


Figure 10: Foot dataset (10,010 points). The first column shows the point set. The second and third columns show two views of the model with boundary triangles depicted in blue.

camera are reconstructed with higher resolution and vice-versa. Given this non-uniform sampling, we use a small neighborhood parameter k to recover parts of the surface with few representative points. However, as using a small neighborhood is less robust to outliers, we need to enforce the splats to have greater support through increasing parameter δ_m. Figure 15(c,d) illustrate the reconstructed surfaces when using a small support ($k = 50$, $\delta_m = 25$): our method is able to reconstruct parts at the back of the scene but also creates many wrong small surfaces at the front (the church building), where the point set contains many structured outliers. Figure 15(e,f) illustrate how our method achieves better robustness when enlarging the support of the splats (same k, but $\delta_m = 40$) at the price of fewer parts of the scene covered in the reconstruction. This suggests that the two parameters k and δ_m are our means to trade robustness for coverage. An automatic parameter selection for adjusting the neighborhood k to the local sampling density is left for future work. Finally, note that since our method seeks a smooth surface, the sharp creases of buildings are not accurately reconstructed.

7. Conclusions and Future Work

We have presented a surface reconstruction method that is able to recover smooth representations of a surface under a large quantity of outliers and noise. The separation of the splat creation and the surface meshing steps makes the method modular, and provides re-usability of the results obtained at each intermediate step. Our method works without any other information than the raw point set, and the local nature of the individual splats makes the method recover, to some extent, the boundaries of the reconstructed shape.

Our approach has some limitations: The reconstructed surfaces are limited to be smooth and may be non-manifold as the surface meshing does not enforce generating manifold surfaces. Our method is not able to reconstruct largely undersampled parts, resulting in holes in areas with missing data. A uniform k parameter is clearly not sufficient when dealing with widely non-uniform point sets.

Future work includes an automatic adaptive selection for k, and a parallelization of all the steps for modern multi-core architectures.

8. Acknowledgements

The authors wish to thank the Stanford 3D Scanning Repository (Stanford University Computer Graphics Laboratory), Michael Kazhdan (Johns Hopkins University), the AIM@SHAPE consortium (the ISTI-CNR Visual Computing Laboratory and Inria), Renaud Keriven and Jean-Philippe Pons for providing the models used in this paper. This work was partially funded through MINECO under grant CTM2010-15216, the EU under grant FP7-ICT-2011-7-288704 and the European Research Council (ERC Starting Grant "Robust Geometry Processing", Grant agreement 257474).

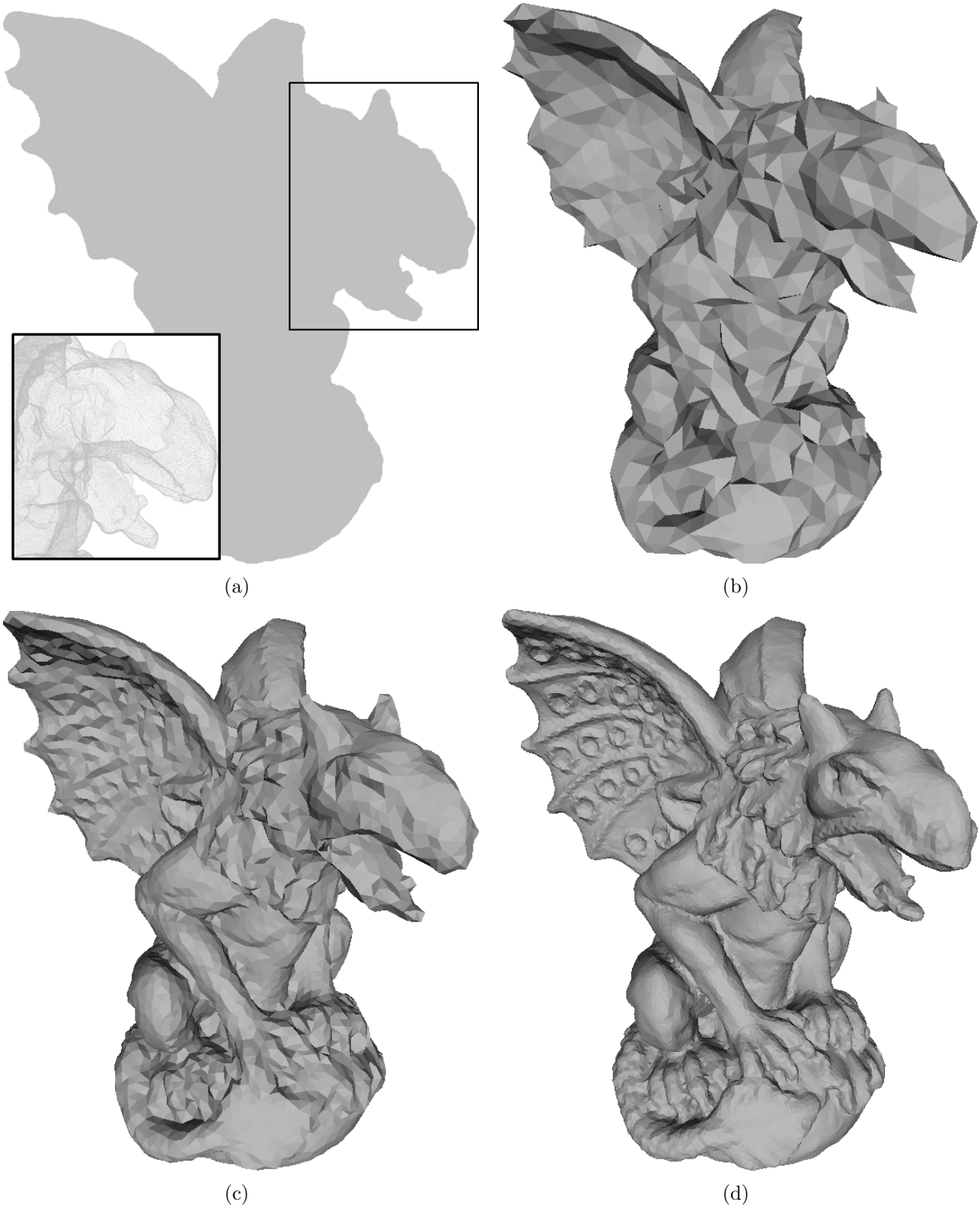


Figure 11: Different resolutions and quality of the final surface can be achieved by changing the parameters of the surface meshing without changing the splats approximation. (a) presents the point set (863, 210 points), while (b), (c) and (d) show incremental levels of resolution with respect to the size of the triangles and the approximation bounds.

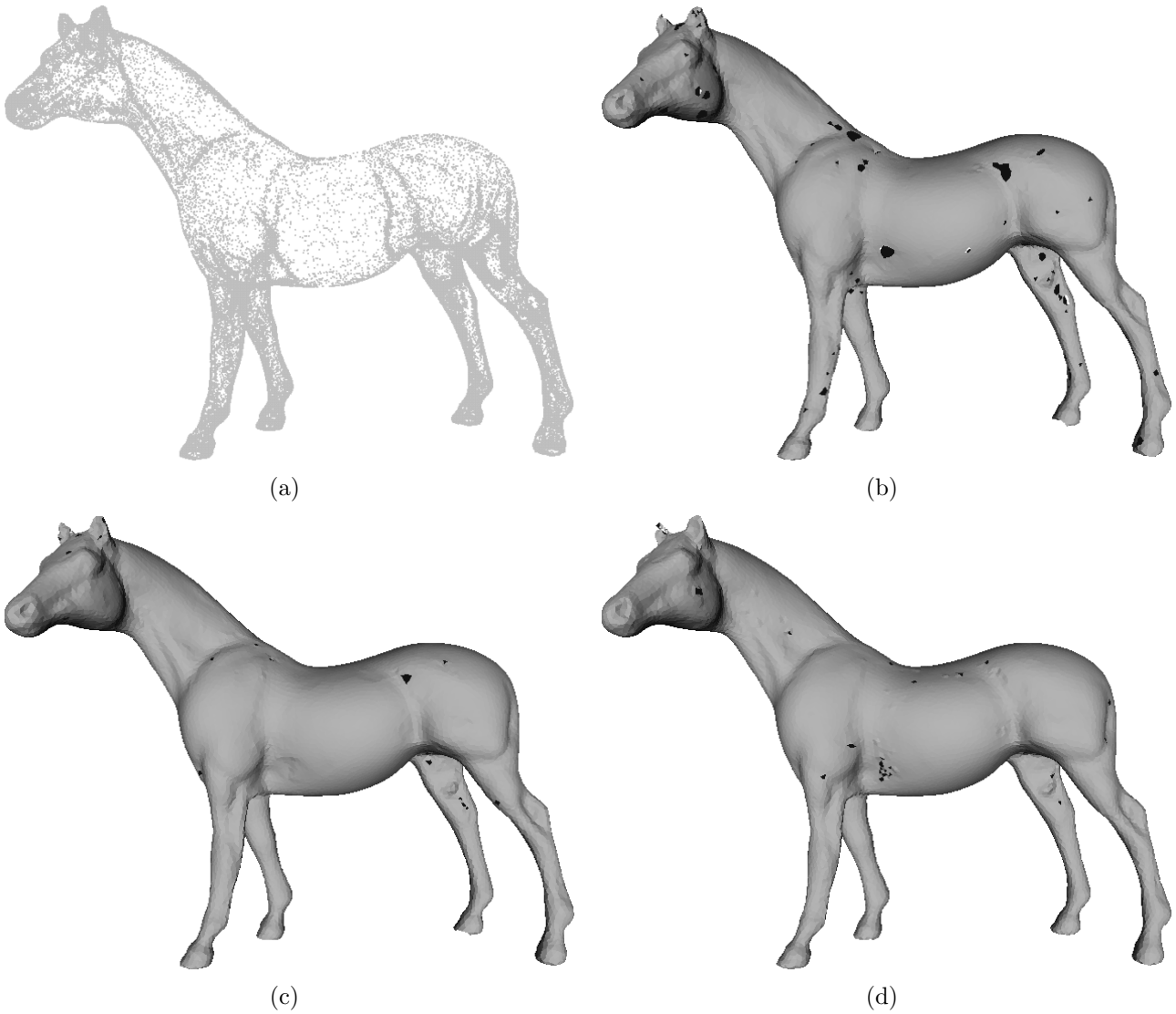
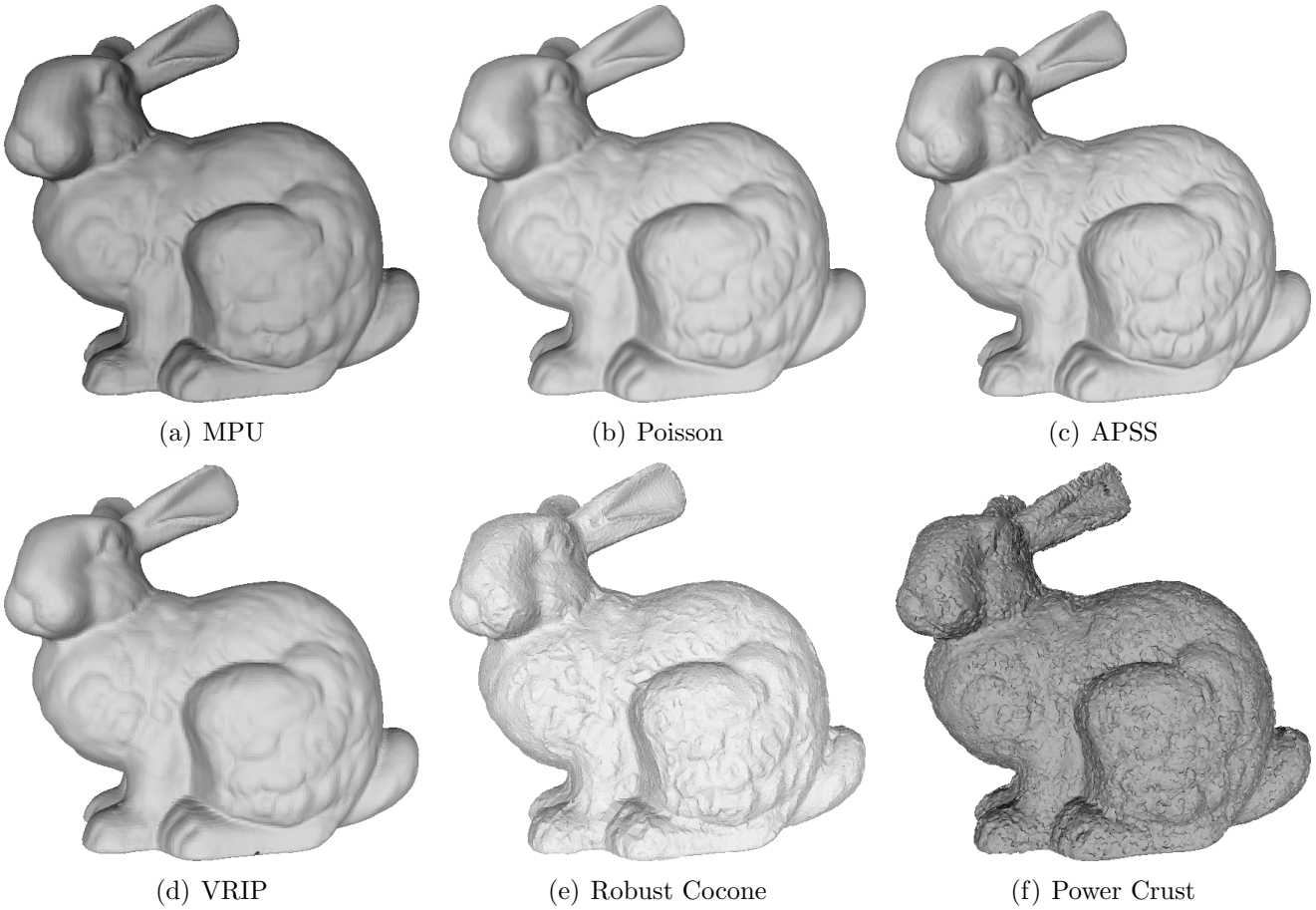


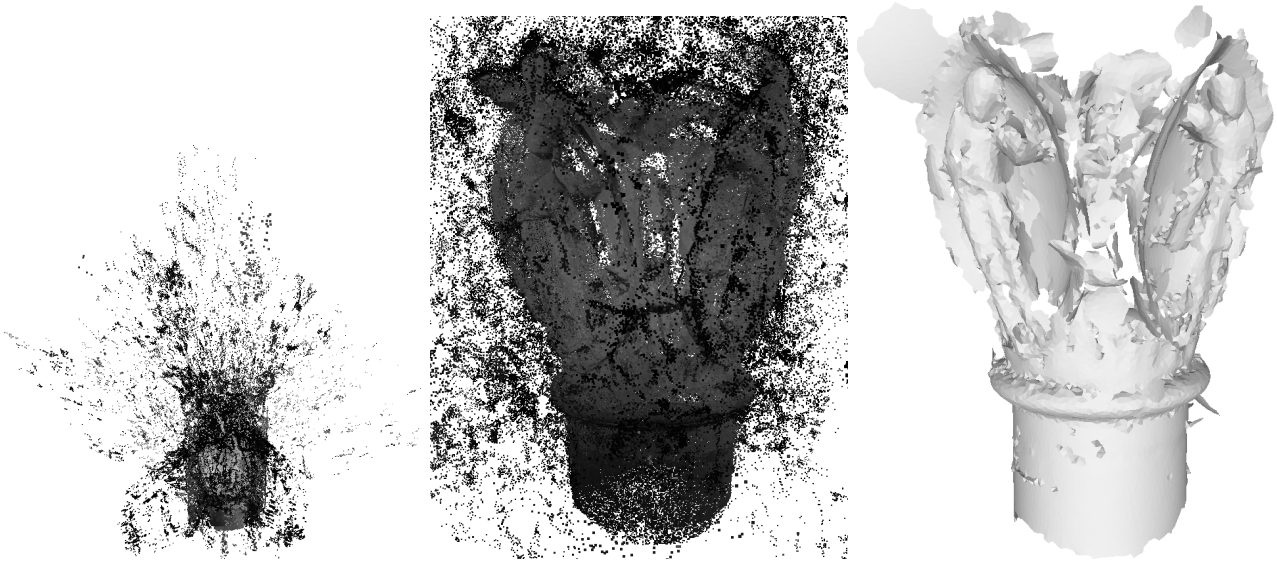
Figure 12: Horse dataset, (a) example of a very sparsely sampled point set, 100,000 points. We then show the results of our method for different values of the k parameter. (b) corresponds to $k = 25$, (c) to $k = 50$ and (d) to $k = 100$.



(a) MPU (b) Poisson (c) APSS

(d) VRIP (e) Robust Cocone (f) Power Crust

Figure 13: Reconstructions for the Stanford bunny dataset using some of the methods from the state of the art. (a) Multi-level Partition of Unity [36], (b) Poisson [14], (c) Algebraic Point Set Surfaces [17], (d) Volumetric Range Image Processor [9], (e) Robust Cocone [35] and (f) Power Crust [2].



(a) (b) (c)

Figure 14: Column Capital optical dataset, 419,488 colored points. (a) Shows the full set of input points, (b) a closeup of the inlier area and (c) the surface obtained by our method.

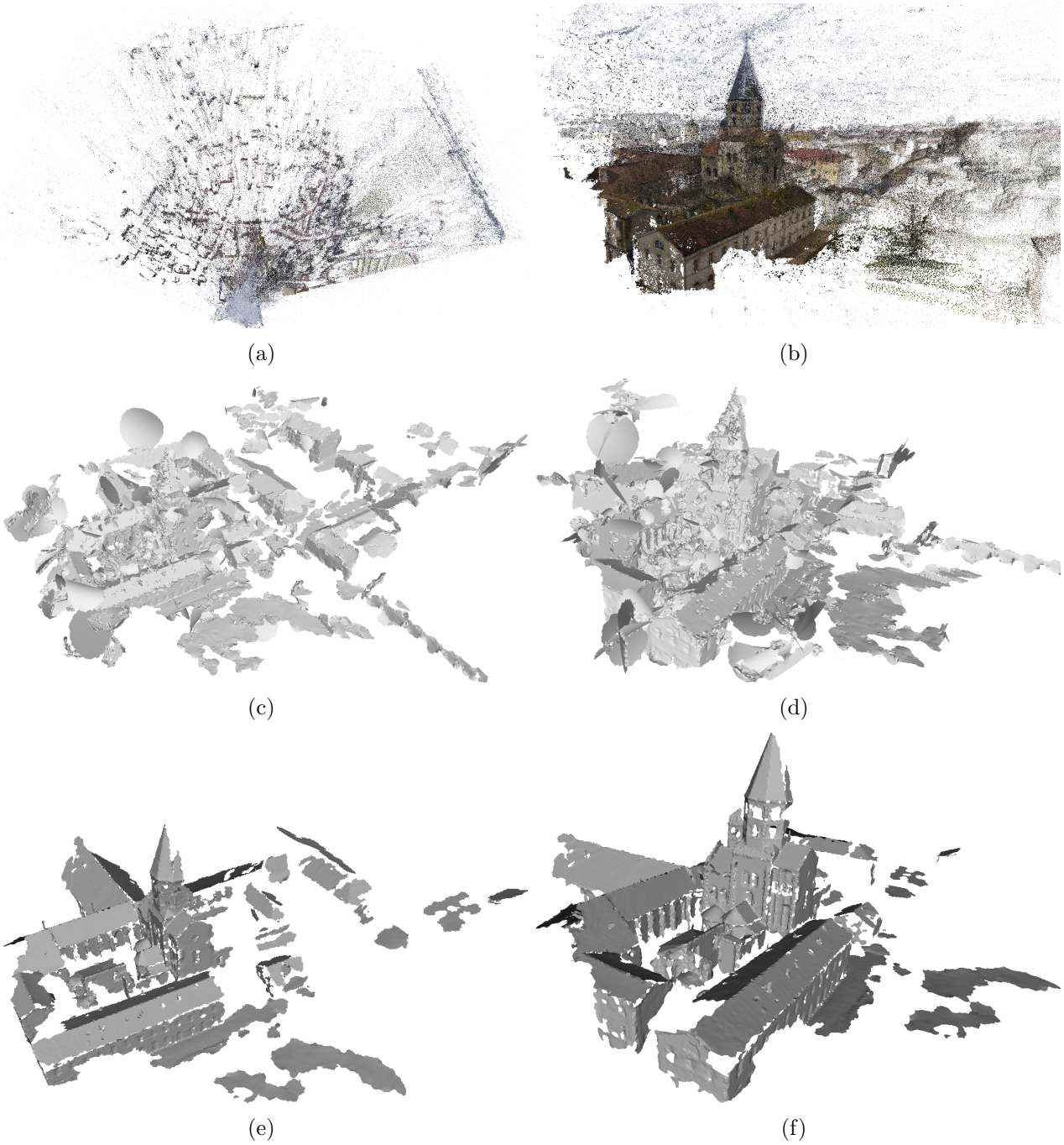


Figure 15: Cluny dataset. (a) Top view of input point set (987,190 colored points with noise and many structured outliers, obtained through aerial photography and dense photogrammetry). (b) Slanted view of input point set. The high anisotropy in sampling density is mainly caused by the camera's position. Buildings closer to the camera, such as the church, are densely sampled, while the houses in the rear part are more sparsely sampled. (c) and (d); Reconstructed surfaces with a small value for parameter δ_m ($k = 50$, $\delta_m = 25$). (e) and (f); Reconstructed surfaces with a larger value for parameter δ_m ($k = 50$, $\delta_m = 40$).

Splats								
Figure	5	7(a)	7(b)	7(c)	7(d)	8r2c1	8r2c2	8r2c3
k	100	50	50	50	50	100	100	100
δ_r (BBD)	0.005	0.01	0.01	0.01	0.01	0.005	0.005	0.005
δ_m	50	15	15	15	15	75	75	75
Run time	412.92	43.96	76.79	164.75	599.70	3919.69	9684.95	11695.6
Figure	8r2c4	8r3c1	8r3c2	8r3c3	8r3c4	9	10	11(b)
k	100	100	100	100	100	100	25	100
δ_r (BBD)	0.005	0.0025	0.0025	0.0025	0.0025	0.015	0.005	0.005
δ_m	75	75	75	75	75	50	15	50
Run time	15339.2	4975.37	10637	15369.8	19925.1	-	2.31	261.99
Figure	11(c)	11(d)	12(b)	12(c)	12(d)	14(c)	15(c)(d)	15(e)(f)
k	100	100	25	50	100	250	50	50
δ_r (BBD)	0.005	0.005	0.005	0.005	0.005	0.0002	0.00003	0.00003
δ_m	50	50	15	25	25	150	25	40
Run time	261.99	261.99	17.46	22.68	35.22	18309.7	10083.2	9987.67
Meshing								
Figure	5	7(a)	7(b)	7(c)	7(d)	8r2c1	8r2c2	8r2c3
α_r/α_d (BBD)	0.0025	0.003	0.003	0.003	0.003	0.003	0.003	0.003
Run time	211.19	43.83	50.03	69.84	74.15	94.49	118.42	98.527
Figure	8r2c4	8r3c1	8r3c2	8r3c3	8r3c4	9	10	11(b)
α_r/α_d (BBD)	0.003	0.003	0.003	0.003	0.003	0.028	0.001	0.02
Run time	107.94	109.74	116.35	114.04	114.72	-	13.48	15.32
Figure	11(c)	11(d)	12(b)	12(c)	12(d)	14(c)	15(c)(d)	15(e)(f)
α_r/α_d (BBD)	0.01	0.005	0.005	0.005	0.005	0.001	0.0002	0.0002
Run time	30.46	84.13	10.54	17.46	30.8	82.66	214.63	49.09

Table 3: Values of the parameters and running times used in each of the presented figures, r stands for row and c stands for column. Values on rows δ_r and α_r/α_d multiply the BBD of the input point set. All computations were performed on an Intel Core i7-2600 CPU @ 3.40GHz, 16Gb RAM. Running times are in seconds. Note that Figure 9 does not have running times since the parameters shown were used on all the corrupted sphere datasets.

9. References

[1] N. Amenta, S. Choi, T. K. Dey, N. Leekha, A simple algorithm for homeomorphic surface reconstruction, in: Proceedings of the sixteenth annual symposium on Computational geometry, SCG'00, ACM, New York, NY, USA, 2000, pp. 213–222.

[2] N. Amenta, S. Choi, R. Kolluri, The power crust, in: Proceedings of the sixth ACM symposium on Solid modeling and applications, SMA'01, ACM, New York, NY, USA, 2001, pp. 249–266.

[3] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin, The ball-pivoting algorithm for surface reconstruction, IEEE Transactions on Visualization and Computer Graphics 5 (4) (1999) 349–359.

[4] J. Digne, J.-M. Morel, C.-M. Souzani, C. Lartigue, Scale space meshing of raw data point sets, Computer Graphics Forum 30 (6) (2011) 1630–1642.

[5] R. Kolluri, J. R. Shewchuk, J. F. O'Brien, Spectral surface reconstruction from noisy point clouds, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, SGP'04, ACM, New York, NY, USA, 2004, pp. 11–21.

[6] P. Labatut, J.-P. Pons, R. Keriven, Robust and efficient surface reconstruction from range data, Computer Graphics Forum.

[7] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, SIGGRAPH Comput. Graph. 21 (1987) 163–169.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, SIGGRAPH Comput. Graph. 26 (2) (1992) 71–78.

[9] B. Curless, M. Levoy, A volumetric method for building complex models from range images, in: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, SIGGRAPH '96, ACM, New York, NY, USA, 1996, pp. 303–312.

[10] H.-K. Zhao, S. Osher, R. Fedkiw, Fast surface reconstruction using the level set method, in: Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01), VLSM'01, IEEE Computer Society, Washington, DC, USA, 2001, p. 194.

[11] A. Hornung, L. Kobbelt, Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information, in: Proceedings of the fourth Eurographics symposium on Geometry processing, SGP'06, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2006, pp. 41–50.

[12] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans, 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, ACM, New York, NY, USA, 2001, pp. 67–76.

[13] Y. Ohtake, A. Belyaev, H.-P. Seidel, 3D scattered data approximation with adaptive compactly supported radial basis functions, in: Proceedings of the Shape Modeling International 2004, IEEE Computer Society, Washington, DC, USA, 2004, pp. 31–39.

[14] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proceedings of the fourth Eurographics symposium on Geometry processing, SGP'06, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2006, pp. 61–70.

[15] D. Levin, Mesh-independent surface interpolation, Geometric Modeling for Scientific Visualization

(2003) 37–49.

[16] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. T. Silva, Computing and rendering point set surfaces, IEEE Transactions on Visualization and Computer Graphics 9 (1) (2003) 3–15.

[17] G. Guennebaud, M. Gross, Algebraic point set surfaces, ACM Trans.Graph. 26 (3) (2007) 23.1–23.9.

[18] A. C. Oztireli, G. Guennebaud, M. Gross, Feature preserving point set surfaces based on non-linear kernel regression, Computer Graphics Forum 28 (2) (2009) 493–501.

[19] P. Mullen, F. D. Goes, M. Desbrun, D. Cohen-Steiner, P. Alliez, Signing the unsigned: Robust surface reconstruction from raw pointsets, Computer Graphics Forum 29 (5) (2010) 1733–1741.

[20] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395.

[21] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for point-cloud shape detection, Computer Graphics Forum 26 (2) (2007) 214–226.

[22] N. J. Mitra, A. Nguyen, L. Guibas, Estimating surface normals in noisy point cloud data, in: special issue of International Journal of Computational Geometry and Applications, Vol. 14(4-5), 2004, pp. 261–276.

[23] N. Amenta, M. Bern, Surface reconstruction by voronoi filtering, in: Proceedings of the fourteenth annual symposium on Computational geometry, SCG'98, ACM, New York, NY, USA, 1998, pp. 39–48.

[24] T. K. Dey, J. Sun, An adaptive mls surface for reconstruction with guarantees, in: Proceedings of the third EUROGRAPHICS symposium on Geometry processing, SGP'05, EUROGRAPHICS Association, 2005, pp. 43–52.

[25] P. Alliez, D. Cohen-Steiner, Y. Tong, M. Desbrun, Voronoi-based variational reconstruction of unoriented point sets, in: Proceedings of the fifth Eurographics symposium on Geometry processing, SGP'07, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2007, pp. 39–48.

[26] B. Li, R. Schnabel, R. Klein, Z. Cheng, G. Dang, J. Shiyao, Robust normal estimation for point clouds with sharp features, Computers & Graphics 34 (2) (2010) 94–106.

[27] J.-D. Boissonnat, S. Oudot, Provably good sampling and meshing of surfaces, Graph. Models 67 (5) (2005) 405–451.

[28] F. Cazals, M. Pouget, Estimating differential quantities using polynomial fitting of osculating jets, in: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, SGP '03, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2003, pp. 177–187.

[29] F. Cazals, M. Pouget, Jet_fitting_3: A Generic C++ Package for Estimating the Differential Properties on Sampled Surfaces via Polynomial Fitting, Research Report RR-6093, INRIA (2007).

[30] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. T. Silva, Point set surfaces, in: Proceedings of the conference on Visualization '01, VIS '01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 21–28.

[31] R. I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, ISBN: 0521540518, 2004.

[32] N. Salman, M. Yvinec, Surface reconstruction from multi-view stereo, in: Proceedings of the 9th Asian Conference on Computer Vision, 2009.

[33] C. A. Lindley, Practical ray tracing in C, John Wiley & Sons, Inc., New York, NY, USA, 1992.

[34] CGAL, Computational Geometry Algorithms Library, http://www.cgal.org.

[35] T. K. Dey, S. Goswami, Provable surface reconstruction from noisy samples, Comput. Geom. Theory Appl. 35 (1) (2006) 124–141.

[36] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, H.-P. Seidel, Multi-level partition of unity implicits, ACM Trans.Graph. 22 (3) (2003) 463–470.