A Point Sampling Algorithm for 3D Matching of Irregular Geometries

Tolga Birdal[1,2] and Slobodan Ilic[1,2]

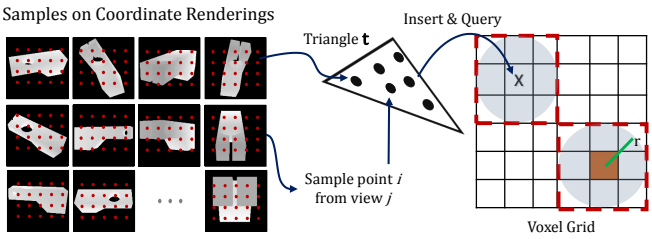
***Abstract*— We present a 3D mesh re-sampling algorithm, carefully tailored for 3D object detection using point pair features (PPF). Computing a sparse representation of objects is critical for the success of state-of-the-art object detection, recognition and pose estimation methods. Yet, sparsity needs to preserve fidelity. To this end, we develop a simple, yet very effective point sampling strategy for detection of any CAD model through geometric hashing. Our approach relies on rendering the object coordinates from a set of views evenly distributed on a sphere. Actual sampling takes place on 2D domain over these renderings; the resulting samples are efficiently merged in 3D with the aid of a special voxel structure and relaxed with Lloyd iterations. The generated vertices are not concentrated only on critical points, as in many keypoint extraction algorithms, and there is even spacing between selected vertices. This is valuable for quantization based detection methods, such as geometric hashing of point pair features. The algorithm is fast and can easily handle the elongated/acute triangles and sharp edges typically existent in industrial CAD models, while automatically pruning the invisible structures. We do not introduce structural changes such as smoothing or interpolation and sample the normals on the original CAD model, achieving the maximum fidelity. We demonstrate the strength of this approach on 3D object detection in comparison to similar sampling algorithms.**

I. Introduction

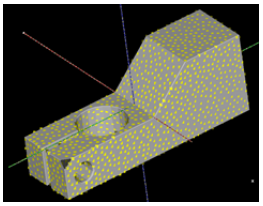
Detection and 6DOF pose estimation of 3D CAD models have ubiquitous use in robotics and industrial applications. Typical CAD models are designed to be parametric forms, where the surface is represented by a collection of mathematical constructs. These constructs are assembled together to compose the complete models. While this is the standard mode of operation in manufacturing, other fields, such as computer graphics, post-process and discretize these models to 3D meshes for resource efficient handling. This discretization is called mesh generation [1] and typically introduces non-uniform, anisotropic and elongated triangles, as well as sharp edges and vertices clustered around feature-rich areas.

State of the art object/scene perception techniques treat the input model as a well behaving point cloud [2], [3], [4], [5]. While the research in keypoint extraction is numerous [6], [7], [5], these methods already operate on a reasonably well distributed vertex set, covering the entire object surface. These render the aforementioned form of discretization very unfriendly for computer vision.

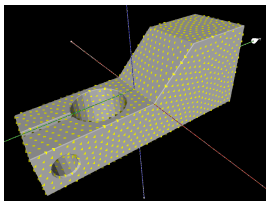
In this paper, we address the particular problem of generating vision-compatible 3D point representations from irregular, non-conforming mesh geometries, amenable to object detection and registration. We contribute by designing a point resampling algorithm, aiding the Geometric Hashing framework of Drost et. al. [2] and Birdal and Ilic [3]. Outcome of our method could certainly be used in other detection pipelines, but we particularly address this one because it inspired a broad range of research and is very practical due to the capability of handling raw point clouds. Their PPF matching relies on uniformly quantizing surfel features. Quantized values act like visual words and are used in retrieval of the model pose. This quantization step is shown to be the most critical source of error [8]. While Birdal and Ilic [9] tackle such problems via soft quantization, they do not devise methods for keypoint selection. With our method, we address the even sampling of both vertices and surface normals resulting in improved detection performance.



(a) Pipeline for initiating samples.



(b) Initialization.



(c) Lloyd relaxation.

Fig. 1: Samples generation. See text for details.[1]

Our input is a 3D mesh. This is the modality we will refer as *CAD model*. Our method starts by presenting a multi-view rendering strategy to form an oversampling of the visible model part. Such sampling is computed by casting rays from all the pixels of all the views and intersecting them with the surface. This can be implemented efficiently by bounding volume hierarchies. A major challenge then stands out to globally unify the sampled views. One of our contributions is to prune and fuse/merge together the points in an effective way to create a bias-free, sparser output. We achieve this by using shallow trees for representing voxel grids. While, at this stage, desired output characteristics can be imposed by the practitioner, we specifically choose to constrain the minimum distances between randomly distributed samples

[1]Department of Computer Aided Medical Procedures, Technical University of Munich, Bolzmannstrasse 3, Munich, Germany.
[2]Siemens AG, Otto Hahn Ring 6, Munich, Germany

[1]CAD Model from MVTec Halcon: `http://www.mvtec.com`

(a.k.a. Poisson Disk Sampling) and the distribution of normals (a.k.a. Normal Space Sampling - NSS) because of the quantization necessity of PPF based geometric hashing [2]. With that, we are able to achieve bias-reduced blue noise (white noise with even spacing) characteristics, which is appealing for this and also other applications [10]. Moreover, for more regularity in the output, we employ a restricted Lloyd relaxation, in which the average disk radius is increased iteratively. With the introduction of this relaxation, blue noise characteristics can be traded-off to distant samples and regular structures. See Fig. 1 for a brief summary.

Main contributions of this work are summarized as:

- We develop a mesh resampling method applicable to any mesh, regardless of the triangles being large, small, acute or elongated.
- We integrate view rendering to bias-reduced sample generation in order to gracefully remove the hidden/invisible geometries.
- We introduce an efficient sparse voxel based algorithm to address the global distribution of resulting vertices and normals using *Poisson Disk Sampling* and *Normal Space Sampling*, in order to satisfy the requirements of PPF matching. We suggest to use the restricted Lloyd relaxation to balance the regularity and randomness.
- We present a GPU implementation for the most computationally heavy part of our algorithm.

Qualitative evaluations and spectral analysis show that we could generate visually appealing samplings with good theoretical properties. Quantitative assessments demonstrate that our algorithm can significantly boost object detection tasks. Our supplementary video can be viewed under `https://youtu.be/uQo535jQ52s`.

II. Prior Art

Sampling is mostly explored in the contexts of mesh to mesh (re-meshing), mesh to points and points to points.

a) Re-meshing: In computer graphics, the state of the art to generate more suitable discrete representations is through re-meshing [11], [12], where a better mesh is obtained in terms of vertex sampling, regularity and triangle quality. A large body of the works in this category are variational and use on Voronoi diagrams. It might be possible that these algorithms create erroneous samples to satisfy the structural penalties. It is also probable that the sharp features are not preserved [11]. Some works explicitly address this [13], sacrificing some mesh quality. On the runtime aspect, many re-meshing algorithms easily reach minutes, making them an overkill, when only point samples are desired. Moreover, naive re-meshing cannot distinguish invisible structures and is thus sub-optimal for vision tasks.

b) Sampling Point Clouds: Rusinkiewicz et.al. propose NSS, for reducing the size of the point clouds for ICP registration [14]. Gelfand et.al. [15] sample the points such that the rigid transformation is constrained the most. Their effective scheme also explicitly addresses ICP. This, by construction, is application specific and tends to select points concentrated around salient regions, whereas our method could be useful in more applications than PPF matching. Another line of research in this category follows sampling points on which good descriptors can be extracted [4], [5]. These methods do not use meshes, cannot benefit from the triangle structure and have to rely on the vertex distribution of original cloud. Tuzel et. al. [16] propose to learn to weigh the model samples per scene. Yet, we are seeking a generic approach, rather than one with a scene specific training.

c) Re-sampling Meshes: Rodola et.al [17] devise a relevance based scheme to sample meshes. *Relevance* is similar to curvature and obviously prone to noise. Along with many other keypoint selectors on meshes [18], [19], [20], they rely on dense distribution of vertices and cannot handle irregular triangles or invisible structures. Descriptor oriented keypoint detection schemes, which are vertex-based [4], [21], [22] remain to be local and do not ensure a global distribution of the samples over the mesh. This doesn't play well with the global or semi-global object modeling, such as geometric hashing of PPF. Moreover, none of these approaches take into account the specific nature of the object detector, as we do in this paper. Birdal et. al. [23] deal with irregularities and visibility issues, but propose a 2D sampling for registration to images. This doesn't generalize to 3D.

Several computer vision scholars, working with mesh models, faced the difficulty we address in this paper. Johnson proposed spin images as local feature descriptors for meshes [24]. He describes a custom re-meshing to aid the spin image computation. Mian et. al. [25] use CAD model meshes under a recognition task and proposes to first render the depth images from different views and then to carry out a volumetric fusion of 3D SDFs using VripPack [26]. This is similar to performing a variant of marching cubes [27]. Note that, this method is limited by voxel resolution and it is expected not to be as quick as the proposed technique. Nonetheless, they do not suffer from the visibility problems, but, unfortunately, authors did not evaluate this aspect of their method. Birdal and Ilic [9] use remeshing to prepare prior models for their reconstruction pipeline.

Point sampling is a long gone study in computer graphics. There, a vast literature exists on sampling with blue noise character for e.g. ray tracing, halftoning or stippling [28]. A famous work with desirable spatial uniformity and absence of artifacts is Poisson Disk Sampling (PDS) [29], [30]. Our work carries traits from PDS and fuses it with the demands of object modeling for detection, resulting in a new algorithm, carrying the good parts of both worlds: Even vertex and normal distribution, graceful handling of irregular or hidden structures and improving the object pose estimation.

III. Method

Before delving into the details, we briefly review the simultaneous object detection and pose estimation of [2].

A. Object Detection via PPF

Given an input mesh, the surface model is described as a database of point pair features $\mathbf{F} = \{\mathbf{f}_{ij}\}$ such that:

$$\mathbf{f}_{ij} = (\|\mathbf{d}\|_2, \angle(\mathbf{n}_i, \mathbf{d}), \angle(\mathbf{n}_j, \mathbf{d}), \angle(\mathbf{n}_i, \mathbf{n}_j)) \qquad (1)$$

where $\mathbf{d}$ is the difference vector, $\mathbf{n}_i$ and $\mathbf{n}_j$ are the surface normals at points $\mathbf{m}_i$ and $\mathbf{m}_j$. $\|\cdot\|$ is the Euclidean distance between point pairs. i refers to the reference point, and j denotes any other paired point. A database is maintained as an inverted file $\mathbf{H}$, where self similar point pair features (PPF) of the 3D model are clustered to belong to the same bucket. Buckets store the indices of reference points, as well as the local reference frame of the point pair. This description is semi-global, in the sense that a reference point is described by the set of other points it is paired with. The creation of a database key, from a given feature is done by quantization:

$$\hat{\mathbf{f}}_{ij} = (\tau\|\mathbf{d}\|_2, \alpha\angle(\mathbf{n}_i, \mathbf{d}), \alpha\angle(\mathbf{n}_j, \mathbf{d}), \alpha\angle(\mathbf{n}_i, \mathbf{n}_j)) \quad (2)$$

where τ and α are relative quantization coefficients of distances and angles, respectively. It is noticeable from Eq. 2 that to populate the hashtable as uniformly as possible, and to increase the discriminative power, an even sampling of both distances and angles is a necessity. This motivates us to develop a tailored point resampling algorithm.

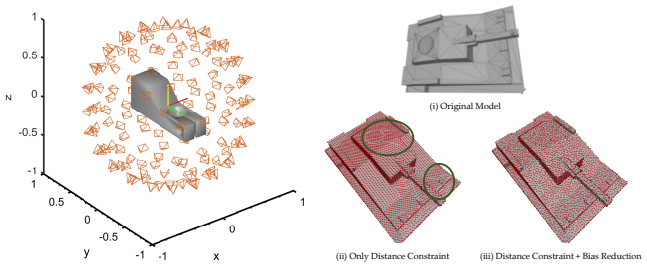
B. Proposed Algorithm

Given an object model, we generate a set of synthetic cameras on a sphere encapsulating the object, as in Fig. 2(a) and cast rays for each view, from the camera center towards the origin. Each ray intersects the mesh, and creates a sample point and a normal at the intersection. We then collect all these samples and prune them using an efficient voxel grid, with Poisson constraints. Finally, with a Llyod relaxation, the sampling gains a balanced regularity. The entire procedure is summarized in Alg. 1 and illustrated in Fig. 1.

Formally, given a mesh model $C = (\mathbf{M}, \mathbf{T})$, with vertices $\mathbf{M} = \{\mathbf{m}_1..\mathbf{m}_{N_m}\} \in \mathbb{R}^{N_m \times 3}$ and triangles $\mathbf{T} = \{\mathbf{t}_1..\mathbf{t}_{N_t}\} \in \mathbb{Z}^{N_t \times 3}$, we aim to generate the point cloud $\mathbf{P} \in \mathbb{R}^{N_p \times 3}$ and normals $\mathbf{N} \in \mathbb{R}^{N_p \times 3}$, s.t. sampled points obey uniform distribution in both 3D space and normal space.

a) Preprocessing: Man-made CAD models might not have design constraints of orientation or positioning and are free to lie anywhere in space. To let the algorithm operate regardless of the model positioning, a first step is to align the model to a canonical reference frame and scale. To do this, we first compute the covariance matrix of the vertices $\mathbf{C}$ and $\mathbf{v}$, the normalized eigen-vectors of $\mathbf{C}$: $\mathbf{C}\mathbf{v} = \lambda\mathbf{v}$. We fix an intermediate reference system as $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_0 \times \mathbf{v}_1)$. The model rotation aligns this new coordinate frame to the base frame $\mathbf{C}_0$. Moreover, the oriented bounding box is computed as the axis aligned bounding box in transformed frame. Even though this approach doesn't necessarily output a volume-minimizing bounding box, it is sufficiently accurate for our purposes. Finally, the mesh is rescaled to the ball with diameter $d = \sqrt{2}$. From here on, with the abuse of notation, the model $\mathbf{M}$ will be referring to this normalized mesh.

b) Multi-View Setting: We generate a set of camera poses (views) $\mathbf{V} = \{V_1...V_N\}$, uniformly distributed on a sphere S, obtained by subdividing the faces of a 12-vertex icosahedron into equally spaced N_s vertices as in Fig. 2(a). A pose is composed of a rotation matrix $\mathbf{R}$ and a translation vector $\boldsymbol{\xi}$: $\mathbf{Q}_i = [\mathbf{R}_i|\boldsymbol{\xi}_i]$, while $\mathbf{o}_i = -\mathbf{R}_i^T\boldsymbol{\xi}_i$ is the camera center. Moreover, for each pose, we maintain a set of intrinsic camera matrices $\mathbb{K} = \{\mathbf{K}_i\}$, according to the pinhole model. Let $\mathbf{f} = (f_x, f_y)$ be the focal length in pixels, and $\mathbf{c} = (c_x, c_y)$, the principal point. We set $c_x = w/2$, $c_y = h/2$, with (w, h), the desired resolution of the camera.



(a) Cam Views (b) Illustration of Sampling Bias

Fig. 2: **(a)** We synthesize camera views around the object as shown. **(b)** Demonstration of Poisson characteristics. Without bias reduction, artifacts caused by view discretization and intersection around the model edges are more visible. To exaggerate the effect, we used 160x120 image resolution.

As we synthesize the camera poses $\mathbf{Q}_i$ from sphere S, we are guaranteed to view the entire projection of the model, but we are not guaranteed to utilize the full resolution, unless $\mathbf{f}$ is tuned. To maximally use the viewport, we first set $\mathbf{f}$ to a relaxed initial value $\mathbf{f} = (f_0, f_0)$, and project the model. Given this projection, we compute a tightly fitting 2D bounding box and scale $\mathbf{f}$ accordingly as:

$$f^* = min\left(\frac{w}{b_w}, \frac{h}{b_h}\right) f_0 \qquad \mathbf{c}^* = 2\mathbf{c} - \mathbf{c}_b \quad (3)$$

where (b_w, b_h) are the dimensions and $\mathbf{c}_b$ is the center of the 2D bounding box. This way, the area of projection is maximized, while preserving the aspect ratio. Because the projected object silhouette is different in all views, $\mathbf{f}^*$ and $\mathbf{c}^*$ differ for each view, resulting in the set $\mathbb{K} = \{\boldsymbol{K}_i\}$.

c) Efficient Ray-Triangle Intersection: In this next stage, the samples projected on the camera views are back-projected and intersected with the 3D mesh itself.

Let $\mathbf{r} = \{\mathbf{o}, \mathbf{d}\}$ denote the ray with origin $\mathbf{o}$ and a normalized direction vector $\mathbf{d}$. Any point on this ray is then parameterized as $\mathbf{r}(\lambda) = \mathbf{o} + \lambda\mathbf{d}$. We then write the edges of the intersecting triangle as $\mathbf{E} = \{\mathbf{e}_o, \mathbf{e}_1, \mathbf{e}_o\}$ and express a triangle by a point and two edges $\mathbf{t} = (\mathbf{v}, \mathbf{e}_0, \mathbf{e}_1)$. The point of intersection $\mathbf{v}_{int}$ can be described using the 2D Barycentric coefficients (u, v) as $\mathbf{v}_{int} = \mathbf{p} + u\mathbf{e}_0 + v\mathbf{e}_1$. Using the famous Möller Trumbore algorithm [31], λ, u and v are obtained. The normal information of the sample p_{int} is then retrieved as the normal of the face: $\mathbf{n}_{int} = \frac{(\mathbf{p}_1 - \mathbf{p}) \times (\mathbf{p}_2 - \mathbf{p})}{\|(\mathbf{p}_1 - \mathbf{p}) \times (\mathbf{p}_2 - \mathbf{p})\|}$.

Note that, since we are using the face normals, we do not have to carry out normal computation for each sample point. Instead, we could pre-compute all the triangle normals and reduce the normal computation to single look-up. If storage is a concern, one could always index the normals during runtime in a hashtable, and ensure single computation per

Algorithm 1 Proposed Sampling Algorithm

Require: Mesh $(\{\mathbf{T}_i\}, \{\mathbf{M}_i\})$, Relative sample threshold τ, Weight threshold τ_w and # Max Samples N_m
Ensure: Sampled point cloud $\mathbf{D}$ with normals $\mathbf{N_D}$
  Normalize and Align $\mathbf{M}$ to canonical frame as in Section IIIa
  Generate camera poses : $\mathbf{V} = \{V_1 ... V_N\}$
  $(\mathbf{S}, \mathbf{N}) \leftarrow []$
  for $V_i \in \mathbf{V}$ **do** ▷ Sample pool generation
    Find best $\mathbf{K}$ via Eq. (2)
    Shoot rays: $\mathbf{r}_i(\lambda) \leftarrow \mathbf{o} + \lambda \mathbf{d}_i$
    $\{(\mathbf{v}^i_{int}, \mathbf{n}^i_{int})\} \leftarrow \{\mathbf{r}_i\} \cap (\mathbf{T}, \mathbf{M})$
    Compute $\mathbf{w}(\phi)$ via Eq. (3)
    Exclude vertices with $\mathbf{w}(\phi) < \tau_w$
    $(\mathbf{S}, \mathbf{N}) = (\mathbf{S}, \mathbf{N}) \cup (\{\mathbf{v}^i_{int}\}, \{\mathbf{n}^i_{int}\})$
  end for
  Randomize$(\mathbf{S}, \mathbf{N})$ as explained in Section IIIg
  Compute CDF from Section IIIi.
  $R_d \leftarrow diameter(\mathbf{S})$
  $(\mathbf{D}, \mathbf{N}_D) \leftarrow []$
  $cnt \leftarrow 0$
  for $cnt < N_m$ **do** ▷ Prune and Merge
    $i \leftarrow find(CDF, random(0, N))$ ▷ See section IIIi
    $(\mathbf{s}, \mathbf{n}) \leftarrow (\mathbf{S}_i, \mathbf{N}_i)$
    $d_{min} = \min_{(\mathbf{t} \in \mathbf{D})} |\mathbf{s} - \mathbf{t}|$ ▷ See Section IIIh.
    if $(d_{min} > \tau R_d)$ **then**
      $\mathbf{D} \leftarrow \mathbf{D} \bigcup \mathbf{s}$
      $\mathbf{N}_D \leftarrow \mathbf{N}_D \bigcup \mathbf{n}$
    end if
    $cnt \leftarrow cnt + 1$
  end for
  Apply Lloyd relaxation on $\{\mathbf{D}, \mathbf{N}_D, \mathbf{T}\}$

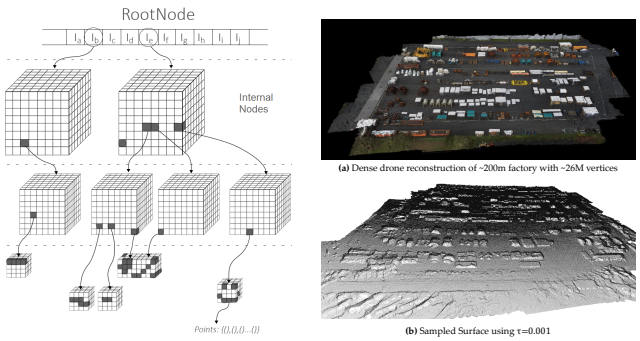
face. The result here is a sample pool $\mathbf{X}$, where CAD model is covered by an over-specified number of points.

d) Weighting Samples: Due to the viewpoint differences and the numerical accuracy of the ray-triangle intersection, not every selected sample has the same quality. Thanks to the ray casting, for each point $\mathbf{m}_j$ and intersecting triangle $\mathbf{t}_i$, we are able to weigh the samples. We first flip the normals $\{\mathbf{n}_i\}$ to point towards the camera, to get $\{\mathbf{n}^v_i\}$. The weight for the sample i is inversely proportional to the angle between the normal and the cast ray and is defined as:

$$w(\phi) = \frac{1}{1 + \exp(-\lambda(\phi - \mu_\phi))} \qquad \phi = 1 - |\mathbf{n}^v_i \mathbf{d}_j| \quad (4)$$

We use $\lambda = 10$ and $\mu_\phi = 0.5$. The maximum weight $w^{max} \approx 1$ is achieved when the vectors $\mathbf{n}^v_i$ and $\mathbf{d}_j$ are parallel ($\phi = 1$), while the minimum $w^{min} \to 0$, is obtained when the angle approaches $90°$ ($\phi = 0$). Note that, due to camera projection, after $\pm 90°$ the face is not viewed, or viewed from the other side. Typically, we reject samples if the weight is found to be very low $w_i < \tau_w$. This lets us to choose the samples which are viewed in a fronto-parallel fashion. This procedure can handle open meshes, as the mesh normals provide directional information.

e) GPU Implementation: For the systems with graphics support, the ray-triangle intersection as well as the weighting can be implemented by rendering the coordinates of intersection on a 3 channel (RGB) image. This can be computed in Shader. Furthermore, the triangle ID per pixel (sample point) can be stored in alpha channel. This reduces the GPU-to-CPU transfer of entire information to a single RGBA texture.



(a) (b)

Fig. 3: (a) Sparse voxel representation. (b) Very large surface mesh and its sampling using the sparse voxels.

f) Merging View Based Samples: Given all the ray intersections, we are left with a set of 3D points $\{\mathbf{p}_i\} \in \mathbf{X}$ per each view, which are to be fused into the full 3D sample cloud $\{p_i\} \in \mathbf{P}$. Some of these points $\mathbf{p}_i$ could be duplicates across views, or even if not, they will be found very close (due to quantization errors). Moreover, a satisfactory distribution of points over the object surface is not yet achieved. Our goal is then to prune this large *sample pool* $\mathbf{X}$, subject to certain constraints. Note that, at this stage, the desired task-based 3D sampling characteristics can be also be enforced. Because typical object detection algorithms [3] rely on equidistant/even sampling, as well as local surface characteristics (such as normals), we adapt two strategies: We employ Poisson disk sampling for distance based constraints, and normal-space sampling to enforce a uniform distribution of local surface characteristics. Our approach fuses these into a single sampling strategy, which we will devise below.

g) Poisson disk sampling (PDS): Recently, PDS is found to be particularly robust for PPF matching [3], due to its generation of even distances and good spectral properties. For our case, it will also help in reducing the bias, caused by discrete sampling on the views. Fig. 2(b) illustrates this effect. PDS tries to obtain uniform random points based on a minimum distance criterion between the samples. Formally, it tries to satisfy the following two conditions:

$$\forall \mathbf{p}_i \in \mathbf{P}_i, \forall \Omega \in D_{i-1}, P(\mathbf{p}_i \in \Omega) = \frac{Area(\Omega)}{Area(D_{i-1})} \quad (5)$$

$$\forall (i, j : i \neq j), \|\mathbf{p}_i - \mathbf{p}_j\| > 2r \quad (6)$$

where D is the domain of the sample pool and D_{i-1} denotes the available (not-yet-sampled) domain. The first condition (Eq. 5 - Poisson Sampling) states a uniformly distributed sample $\mathbf{p}_i$ falls in subdomain Ω with likelihood, proportional to the area of Ω, provided that $D_{i-1} \cap \Omega = \emptyset$. Due to the computational complexity of calculating the region of sampling, we relax the first constraint. We associate each sample $\mathbf{p}_i$ to a sphere centered at $\mathbf{p}_i$ with radius r. r is the same for all the samples. The probability $P(\mathbf{p}_i \in \Omega)$ is then computed as $P(\mathbf{p}_i \in \Omega) = 1 - \sum_{j=1}^{i} \alpha(j) 4/3 \pi r^3$, where $\alpha(j)$ influences a poisson process bound in the interval $(0, 1]$,

influencing the percentage of the feasible sampling space for the point i. Instead of using $\{\alpha(j)\}$ as in a Poisson process, we simply replace it with its expected value $\bar{\alpha}$, reducing to $P(\mathbf{p}_i \in \Omega) = 1 - 4/3K\bar{\alpha}\pi r^3$. By that, we are allowed to retrieve samples from the pool, sequentially with equal likelihood, i.e. the drawing is uniformly distributed.

Satisfying the 2^{nd} condition (Eq. 6 - Disk Sampling) is more trivial but required to be made efficient for a large number of samples. The main idea is to draw samples from the pool $\boldsymbol{X}$ iteratively and check against the existing samples $\mathbf{P}_{i-1}$, for the violation of Eq. 6. If Eq. 6 is satisfied, the sample is accepted. A naive implementation involves a search through all the so-far sampled points or marking all the neighbors of a sample as *rejected*. Both are computationally demanding. We take a different path and construct a 3D voxel grid G over the existing samples. We then take a sample $\mathbf{p}_i$ from the pool sequentially and insert it into G. If the sample satisfies Eq. 6, it is kept, otherwise rejected. The side length of the grid is tuned such that the search ball remains within 9 voxels, and the query can be done in $O(1)$ time, enabling us to complete the entire sampling in $O(N)$. Even though this procedure is greedy - as it depends on the first sampled point- it is found to generate good distribution in practice.

h) Sparse Voxel Representation: For large radii (less samples), the search in dense voxel grids will be fast. However small radii, where closer points are sampled are problematic due to the exploding memory of the grid. For that reason, we propose to use a sparser voxel-grid similar to [32]. Our tree structure resembles the one in a B+ tree. By construction, the tree is height-balanced, shallow and wide. This decreases the number of operations for traversing the tree from root to a leaf node. The structure implemented has 1 root node, 2 internal layers and a leaf layer as shown in Fig. 3(a). Point information is stored only at the leaf layer, wheres the internal layers maintain a bitmask for encoding active children based on their spatial coordinates. To increase traversal speed, the whole structure is restricted to powers of two: First layer has a size of 8 units per dimension, second layer 4 and leaf nodes 4. A *Cache* that holds the last visited internal and leaf nodes is also implemented. The resolution of the grid is automatically adjusted to $2r$. This way, given a point, we can answer the question *should this point be sampled?* in constant time and can therefore downsample very large point clouds such as the one in Fig. 3(b).

i) Normal Space Sampling: Rusinkiewicz and Levoy [14] propose to sample points such that the normals are uniformly distributed. We adapt this into our pipeline by altering the order of the considered points. We first quantize all normals into a dense set of bins $\{\bar{n}_j\}$ and assign each normal to the closest bin. We then take the random ordering specified in Section IIIg, and compute the cumulative distribution function $CDF = F(\bar{n}_i)$. After that, a scalar γ is drawn uniformly in the interval $[0, F(\bar{n}_i)]$ and the smallest index i s.t. $F(\bar{n}_i) > \gamma$ is computed, using binary search. Since this can be done subsequent to the random sampling, it has little effect in introducing regularity, while still satisfying the uniform distribution of normals.

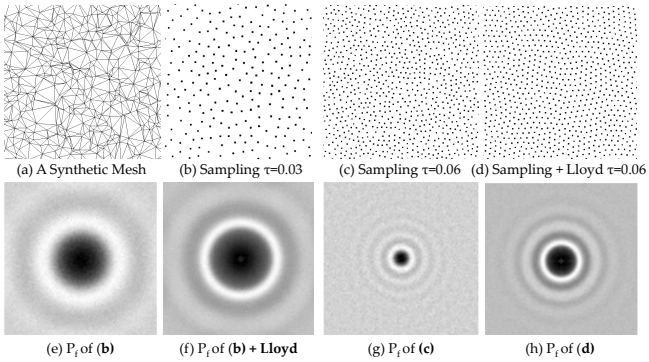


Fig. 4: Spectral analysis. **(a)** Original Mesh. **(b,c)** Sampling with different radii. **(d)** Sampling + Lloyd relaxation. **(e)** Power spectrum plot of (b). **(f)** Spectral plot of (b) after 10 Lloyd steps. **(g)** Spectral plot of (c). **(h)** Spectral plot of (d)

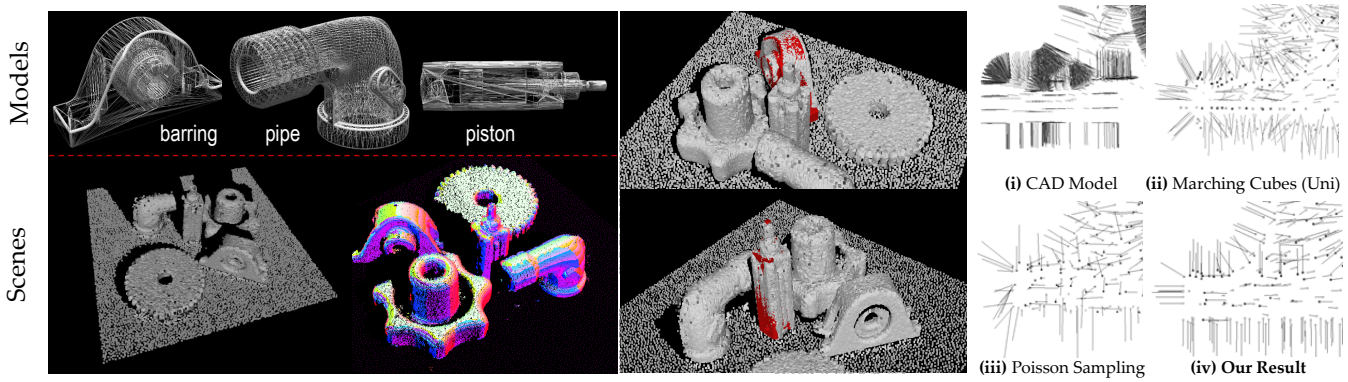
j) Lloyd Relaxation: Due to the introduced randomness and greediness of the sampling algorithm, the sampled points are not regular. To obtain a good balance between blue noise property and regularity, we conclude our sampling by applying a few Llyod iterations, in which the centers of the points are shifted to the centers of the Voronoi diagram, gradually. While, it is easy to apply this on synthesized samples (samples not on a specific surface), ensuring that the Voronoi centers remain on the surface is difficult. For that, we exploit Restricted Centroidal Voronoi (R-CVD) iterations, efficiently implemented by [33]. R-CVD operates by intersecting (restricting) the VD with the surface. Lloyd scheme is formulated as a variational energy minimization and a quasi-newton approximation is made for fast convergence. Because minimum distance constrained is roughly satisfied in the previous section, this scheme enjoys a good initialization and only a few iterations are enough for pleasing results.

IV. Results and Evaluation

We present evaluations of our method as well as a spectral analysis for justifying frequency space properties.

a) Spectral Analysis: Since our sampling exhibits blue noise characteristics, we use frequency domain analysis to evaluate the quality. Lagae and Dutre [34] standardize this analysis as a power spectrum study. First, we generate 50 different synthetic 2D meshes as shown in Fig. 4 and sample them with our method. For each sampling, the periodogram is computed. These periodograms are averaged to estimate the power spectrum $P(f)$. We plot these 2D spectra in Fig. 4 with a logarithmic tone map, removing the high-magnitude DC component. $P(f)$ reveals the typical *blue noise* properties: The central DC peak is surrounded by an annulus of low energy, followed by a sharp transition region, a low-frequency cutoff and a flatter high-frequency region. As a result, inter-sample distances follow a certain power law, with high frequencies being more common. Note that our sampling preserves these spectral properties.

b) Real Dataset and Parameters: As our method best performs with industrial CAD models in mesh forms, we utilize the Toshiba dataset, proposed in [35]. This dataset



(a) Our Dataset **(b)** Typical Detection Results **(c)** Sampled Normals

Fig. 5: **(a)** Views of Toshiba dataset. Upper row: CAD models with anisotropic triangles and internal structures; Lower row: A scene point cloud and its normals. **(b)** Models detected by [2] using our sampling. **(c)** Normals on sample points compared. Notice our sampling (iv) can generate normals which are closest to their original CAD counterparts (i).

consists of real life wireframe 3d models and scanned point clouds as scenes. The models have internal structures, elongated triangles and sharp edges, making them hard to work with. We modify this dataset to include normals for 20 scenes, each scene containing multiple objects at different occlusion levels. The dataset is summarized in Fig 5a. For all the experiments in this paper a set of fixed parameters are used: $\tau = 0.03$; image resolution is 640x480; the weight function/threshold are shown in Fig. 7(a) and we use 2 subdivisions of the sphere, resulting in 162 camera poses viewing the object.

c) Base Methods: We compare our approach against 4 other methods, which are capable of converting meshes into point sampled surfaces. Our baseline is set to be the *Monte Carlo (MC)* sampling (a uniform random sampling) within the facets. A more proper method is the *stratified sampling* or triangle subdivision method [36], in which each triangle gets a sample depending on the resolution of an underlying voxel grid. A mainstream and successful approach is *Poisson disk sampling* [29], also used in [3]. Finally, we compare our method against a *uniform mesh resampling*, which consists of building a uniform volumetric signed distance field and applying a marching cubes to get uniformly distributed samples. This is also very similar to the method proposed in [25]. Note that a large family of 3d keypoint extraction methods operate directly on vertices [4], [37] and cannot be applied to Toshiba dataset or this application.

d) Qualitative Evaluations: We assess the visual quality of our method on Piston object from the Toshiba dataset as well as on two additional industrial CAD models of a gasoline engine and a large transformator. All models have complicated triangle arrangement, hidden faces and structural connections. The vertices are unevenly distributed and clustered on certain support areas. Fig. 6 presents the results of sampling 1000 points by our algorithm and preceding methods. It is noticeable that our method better preserves the global shape, has even distribution of vertices and can get rid of internal structures. Moreover, thanks to the blue noise characteristics, the discretization artifacts due to the views are not visible. These result in an enhanced perceptual quality of the generated point sets. The closest result to ours is from Poisson sampling, but as it doesn't treat hidden faces or surface normals, our method remains visually superior. We also visualize the normals estimated with different techniques as well as ours in Fig. 5c. Here, one should look for the set of directions, which resemble the CAD model's the most. Because we sample directly on the faces, the normal estimation quality of ours is also the closest to the original CAD model among the algorithms we evaluate. Please refer to our suppl. material for further visual results.

e) Application to Object Detection: We now assess the effectiveness of the method in the challenge of 3D detection and pose estimation, the reason of its design. We use the PPF method of [3], [2] as explained earlier, on the Toshiba dataset [35]. The only modification is that, we replace the weighting scheme of [3] with the weights computed in Section III. The evaluations are done by substituting the existing sampling of [2] with a different one and comparing the pose estimation results. Our findings for different number of samples and sampling algorithms are plotted in Fig. 8. We provide distinct plots for the rotational ($\mathcal{E}_{\mathbf{R}}$) and translational ($\mathcal{E}_{\mathbf{t}}$) error components, which are respectively measured as:

$$\mathcal{E}_{\mathbf{R}}(\mathbf{R}_t, \mathbf{R}_g) = \arccos\left(\frac{\text{trace}(\mathbf{R}_t^{-1}\mathbf{R}_g) - 1}{2}\right)\frac{1}{\pi} \quad (7)$$

and $\mathcal{E}_{\mathbf{t}}(\mathbf{t}_t, \mathbf{t}_g) = \|\mathbf{t}_t - \mathbf{t}_g\|$. A detection is said to be correct if it can be refined to the correct ground truth pose by a subsequent ICP (iterative closest point) alignment. Depending on the pose with which the object lands on the scene, the tolerance of ICP could differ and we use the same setting across all methods. We only accumulate the pose errors when the correct detection is spotted. Fig. 5b illustrates these refined poses. We plot the number of correct detection per each sampling in the left-most column of Fig. 8. It is consistently visible from this figure that choosing evenly spaced points on the visible surface with uniform

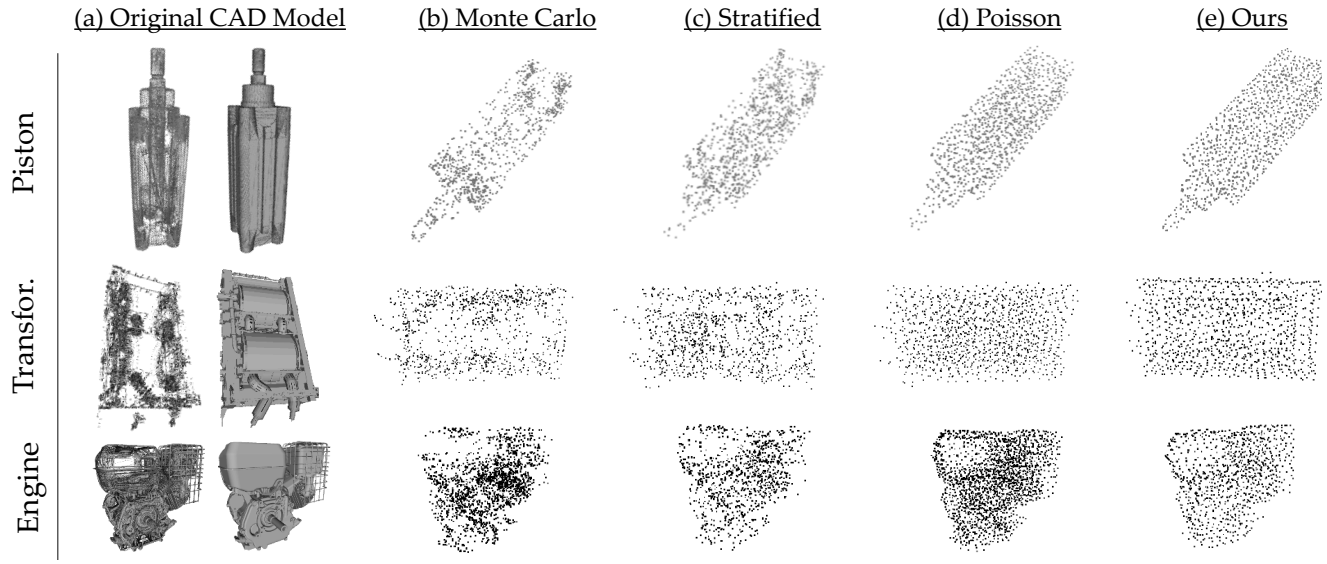
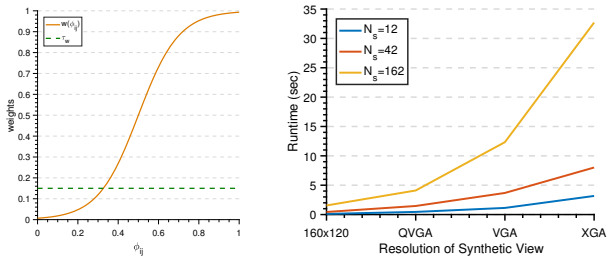


Fig. 6: Visuals of our point resampling. Columns depict different samplings of the same object, while the rows contain different objects. First two columns show the mesh and the vertices of the original models.



(a) (b)

Fig. 7: **(a)** Used weighting function (see Section IIId). **(b)** Computational timings of CPU implementation.

distribution of normals helps our algorithm to detect more objects and make less error in the pose when the object is detected. This means that we manage to sample more task suited points which have higher probability of being seen and have a better coverage of the entire surface - as for [2], the pose depends on the point found to be on the surface.

The SDF based sampling of [25] is the closest to ours in terms of pose estimation performance. However, it is noticed that as the number of samples decrease, the performance gap grows. This is because our samples are always located on visible primitives and therefore are better suited to perception/detection tasks. We have also noticed that due to voxel-grid used in Marching Cubes, the memory requirements of [25] become intractable as the model size increases. Our algorithm, on the other hand, doesn't suffer from this issue and can sample models of arbitrary size (see Fig. 3(b)).

f) Computational Time: We implemented our algorithm on an Intel i5 2.3GhZ CPU. For ray casting, we use the freely available Intel Embree library [38], carefully optimized for Intel platform. The average timings on our dataset are plotted in Fig. 7(b). The most important parameter for runtime is the resolution of the synthetic views as we cast a ray for each pixel of the view. The number of faces of the CAD model has a diminished effect due to the apriori spatial indexing. Our GPU implementation, on the average, could only run twice as fast and is dominated by the transfer overhead of the textures. Note that the view merging is always computed on the CPU, to ease the implementation complexity.

V. Conclusion

In this paper we proposed a very effective, practical and easy to implement point resampling strategy crafted for computer vision problems, in particular PPF matching. Our algorithm allows us to directly compute a randomized, evenly spaced point sampling of the visible portion of the surface. The normal computation and point quality are well integrated into the schema and are always present in the sampled result. We evaluated our approach qualitatively and quantitatively on a real dataset, and demonstrated the boost in the matching and pose estimation tasks. We further like to enlarge the appliance of the method to different problems such as reconstruction and multi-view registration.

References

[1] J. Schöberl, "Netgen an advancing front 2d/3d-mesh generator based on abstract rules," *Computing and Visualization in Science*, 1997.

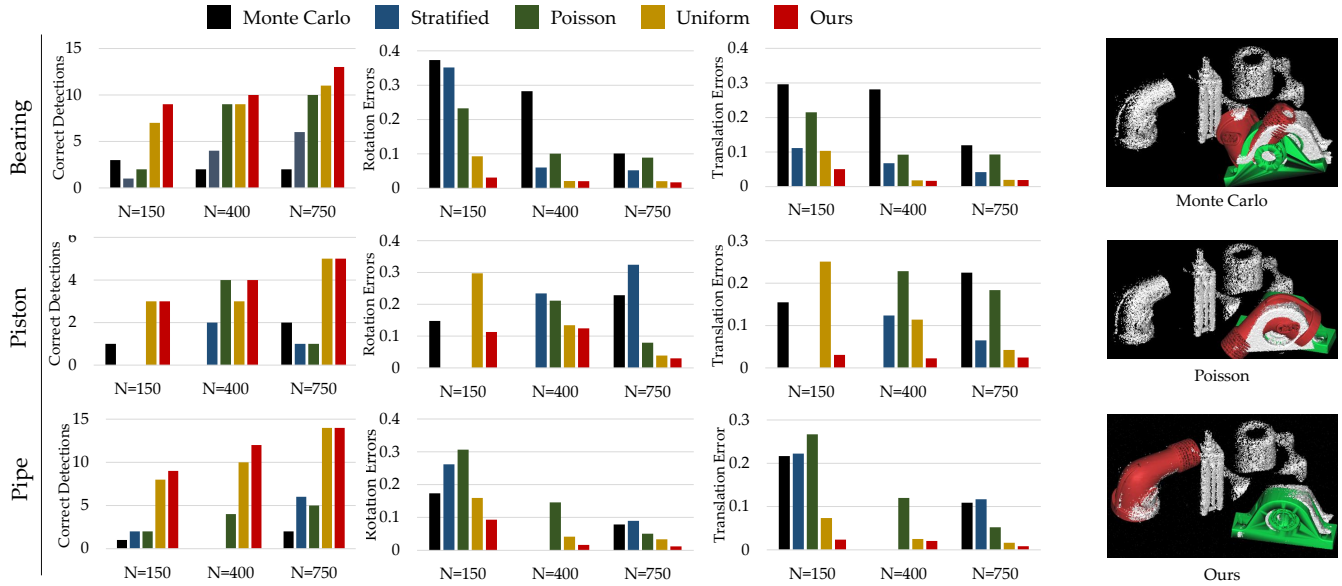
[2] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *Computer Vision and Pattern Recognition*, 2010.

[3] T. Birdal and S. Ilic, "Point pair features based object detection and pose estimation revisited," in *3D Vision*. IEEE, 2015, pp. 527–535.

[4] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3d object recognition," in *International Conference on Computer Vision Workshops*. IEEE, 2009, pp. 689–696.

[5] J. Serafin, E. Olson, and G. Grisetti, "Fast and robust 3d feature extraction from sparse point clouds," in *Intelligent Robots and Systems, International Conference on*. IEEE, 2016.

[6] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European conference on computer vision*. Springer, 2010, pp. 356–369.



(a) Quantitative Results (b) Matching Results

Fig. 8: **(a)** Pose errors, for different number of samples N, computed for correct detections only and are averaged. Our sampling could enable the same detection method to obtain more correct results with lower pose errors, regardless the sample count. **(b)** Scene from Toshiba dataset. Detection results by different sampling algorithms are visualized on this point cloud.

[7] S. M. Prakhya, B. Liu, and W. Lin, "Detecting keypoint sets on 3d point clouds via histogram of normal orientations," *Pattern Recognition Letters*, vol. 83, 2016.

[8] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige, "Going further with point pair features," in *European Conference on Computer Vision*. Springer, 2016, pp. 834–848.

[9] T. Birdal and S. Ilic, "Cad priors for accurate and flexible instance reconstruction," *CoRR*, vol. abs/1705.03111, 2017.

[10] D.-M. Yan, J.-W. Guo, B. Wang, X.-P. Zhang, and P. Wonka, "A survey of blue-noise sampling and its applications," *Journal of Computer Science and Technology*, 2015.

[11] B. Lévy and Y. Liu, "L p centroidal voronoi tessellation and its applications," in *ACM Transactions on Graphics (TOG)*, 2010.

[12] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene, "Recent advances in remeshing of surfaces," in *Shape Analysis and Structuring*, 2008.

[13] J. Vorsatz, C. Rössl, L. P. Kobbelt, and H.-P. Seidel, "Feature sensitive remeshing," in *Computer Graphics Forum*, 2001.

[14] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *3-D Digital Imaging and Modeling*. IEEE, 2001.

[15] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy, "Geometrically stable sampling for the icp algorithm," in *3-D Digital Imaging and Modeling*. IEEE, 2003.

[16] O. Tuzel, M.-Y. Liu, Y. Taguchi, and A. Raghunathan, "Learning to rank 3d features." in *ECCV (1)*, 2014, pp. 520–535.

[17] E. Rodolà, A. Albarelli, D. Cremers, and A. Torsello, "A simple and effective relevance-based point sampling for 3d shapes," *Pattern Recognition Letters*, vol. 59, pp. 41–47, 2015.

[18] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud, "Surface feature detection and description with applications to mesh matching," in *Computer Vision and Pattern Recognition*. IEEE, 2009.

[19] S. Salti, F. Tombari, R. Spezialetti, and L. Di Stefano, "Learning a descriptor-specific 3d keypoint detector," in *ICCV*, 2015.

[20] S. Filipe and L. A. Alexandre, "A comparative evaluation of 3d keypoint detectors in a rgb-d object dataset," in *Computer Vision Theory and Applications*, vol. 1. IEEE, 2014.

[21] F. Tombari, S. Salti, and L. Di Stefano, "Performance evaluation of 3d keypoint detectors," *IJCV*, 2013.

[22] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3d local feature descriptors," *International Journal of Computer Vision*, vol. 116, pp. 66–89, 2016.

[23] T. Birdal, E. Bala, T. Eren, and S. Ilic, "Online inspection of 3d parts via a locally overlapping camera network," in *WACV 2016: IEEE Winter Conference on Applications of Computer Vision*. IEEE, 2016.

[24] A. Johnson, "Spin-images: A representation for 3-d surface matching," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[25] A. S. Mian, M. Bennamoun, and R. Owens, "Three-dimensional model-based object recognition and segmentation in cluttered scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, Oct. 2006.

[26] B. Curless, "Vrippack users guide," 2006.

[27] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *ACM siggraph computer graphics*, vol. 21. ACM, 1987, pp. 163–169.

[28] R. A. Ulichney, "Dithering with blue noise," *IEEE*, vol. 76, 1988.

[29] M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes," *Visualization and Computer Graphics*, vol. 18, pp. 914–924, 2012.

[30] A. G. Ahmed, H. Perrier, D. Coeurjolly, V. Ostromoukhov, J. Guo, D.-M. Yan, H. Huang, and O. Deussen, "Low-discrepancy blue noise sampling," *ACM Transactions on Graphics (TOG)*, vol. 35, 2016.

[31] T. Möller and B. Trumbore, "Fast, minimum storage ray/triangle intersection," in *ACM SIGGRAPH Courses*. ACM, 2005, p. 7.

[32] K. Museth, J. Lait, J. Johanson, J. Budsberg, R. Henderson, M. Alden, P. Cucka, D. Hill, and A. Pearce, "Openvdb: an open-source data structure and toolkit for high-resolution volumes," in *ACM*, 2013.

[33] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, "Isotropic remeshing with fast and exact computation of restricted voronoi diagram," in *Computer graphics forum*, vol. 28, 2009.

[34] A. Lagae and P. Dutré, "A comparison of methods for generating poisson disk distributions," in *Computer Graphics Forum*, 2008.

[35] M.-T. Pham, O. J. Woodford, F. Perbet, A. Maki, B. Stenger, and R. Cipolla, "A new distance for scale-invariant 3D shape recognition and registration," in *ICCV*, 2011.

[36] M. Alexa, S. Rusinkiewicz, D. Nehab, and P. Shilane, "Stratified point sampling of 3d models," in *Eurographics*, 2004.

[37] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3d object recognition in cluttered scenes with local surface features: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, 2014.

[38] S. Woop, L. Feng, I. Wald, and C. Benthin, "Embree ray tracing kernels for cpus and the xeon phi architecture," in *ACM SIGGRAPH 2013 Talks*. ACM, 2013, p. 44.