SCONE: Surface Coverage Optimization in Unknown Environments by Volumetric Integration

Appendix

Antoine Guédon **Pascal Monasse** **Vincent Lepetit**
LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, France
{antoine.guedon,pascal.monasse,vincent.lepetit}@enpc.fr

In this appendix, we provide the following elements:

1. A complete proof of our main theorem 1 in Section A.
2. Further details about the training process, as well as the implementation of SCONE, in Section B.
3. Further details about the experiments we conducted in Section C.

Code and data are available on our dedicated webpage: `https://github.com/Anttwo/SCONE`.

A Proof of Theorem 1

In this section, we keep all notations introduced in the main paper and give further details about the proof of Theorem 1.

In particular, we start by listing all assumptions we make about the scene's geometry and discuss whether they are appropriate or not to real-case scenarios. Then, we present the complete derivations leading to Theorem 1.

A.1 Main Assumptions

Assumption 1: The surface of the scene consists in a finite collection of bounded watertight surfaces. To derive Theorem 1, we need the scene to be represented as a closed volume χ, and its surface as the boundary $\partial\chi$ of χ, which is trivially true for a finite collection of bounded watertight surfaces. Such an assumption is realistic since a real 3D object has a non-zero volume and is actually made of watertight surfaces. Note that, in practice, the floor is scanned as a plane surface by a depth sensor. The same observation applies to walls in indoor scenes, as well as all surfaces that delimit unreachable parts of 3D space (except for the inside of objects). However, these surfaces still delimit volumes in reality, and an arbitrary thickness can be predicted for them, as it won't change the result of the scan.

Following such assumptions, the Signed Distance Function (SDF) $f_\chi : \mathbb{R}^3 \to \mathbb{R}$ of the volume χ is defined as:

$$f_\chi(x) = \begin{cases} d(x, \partial\chi) & \text{if } x \in \chi \\ -d(x, \partial\chi) & \text{if } x \notin \chi \,, \end{cases} \tag{1}$$

where $d(x, \partial\chi) = \inf_{y \in \partial\chi} \|x - y\|_2$ for any $x \in \chi$.

As we never reconstruct infinitely large scenes in practice, we also consider the scene to be bounded. As a consequence, both χ and $\partial\chi$ are compact as they are bounded, closed subsets of $\mathbb{R}^3$.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

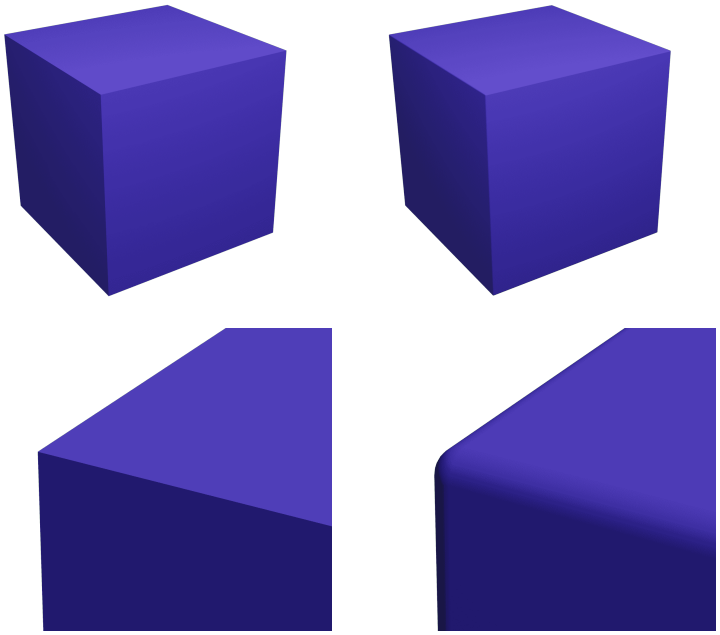


Figure 1: **Example of a C^2 approximation of a surface**. On the left, a cube mesh with 6 faces and 12 sharp edges. On the right, a smoother C^2 approximation of the same cube.

Assumption 2: The boundary $\partial\chi$ is C^2. We suppose the scene's surface is regular enough to be, locally, the image of a C^2 embedding of $\mathbb{R}^2$ into $\mathbb{R}^3$. This is a non-trivial assumption since in theory, the surface of every object with sharp edges or corners is not a C^2 boundary. However, in practice most of real objects with edges (like a door, a table, etc.) are smoother than they appear, and can be accurately represented by a C^2 boundary. In general, we believe that a C^2 approximation of most realistic non-smooth surface is enough to compute a meaningful NBV anyway, as shown in Figure 1.

Following such assumptions, according to [3], there exists a quantity $\mu_0 > 0$ such that the SDF f_χ is twice continuously differentiable on the spherical neighborhood $T(\partial\chi, \mu_0) := \left\{p \in \mathbb{R}^3 \mid \exists x \in \partial\chi, \|x - p\|_2 < \mu_0\right\}$.

In particular, for all $x_0 \in \partial\chi$, the function f_χ satisfies $\nabla f_\chi(x_0) = N(x_0)$ where N is the inward normal vector field [3]. Moreover, for all absolutely integrable function $g : T(\partial\chi, \mu_0) \to \mathbb{R}$, and for all $\mu < \mu_0$, the following identity is satisfied [3]:

$$\int_{T(\partial\chi,\mu)} g(x)\mathrm{d}x = \int_{\partial\chi}\int_{-\mu}^{\mu} g(x_0 + \lambda N(x_0))\ \det(I - \lambda W_{x_0})\,\mathrm{d}\lambda\,\mathrm{d}x_0, \tag{2}$$

where W_{x_0} is the Weingarten map at x_0, that is, the Hessian of f_χ, which is continuous on $T(\partial\chi, \mu_0)$ since f_χ is C^2 on $T(\partial\chi, \mu_0)$. The integral on the left of Equation 2 is a volumetric integral on a spherical neighborhood, while the integral on the right is a surface integral on $\partial\chi$.

Please note that Equation 2 from [3] actually only applies to tubular neighborhoods, which are specific neighborhoods of submanifolds resembling the normal bundle. However, since the spherical neighborhoods of C^2 watertight surfaces also are tubular neighborhoods, we prefer to use the simpler definition of spherical neighborhoods.

A.2 Proof of Theorem 1

To derive the theorem, we first prove the following lemma.

Lemma 2. *Under the previous assumptions, there exists $\lambda_0 > 0$ such that, for all $\lambda < \lambda_0$ and $x_0 \in \partial\chi$, the point $x_0 + \lambda N(x_0)$ is located inside the volume,* i.e.*, $f_\chi(x_0 + \lambda N(x_0)) \geq 0$.*

Proof. Following our main assumptions, f_χ is C^2 on the spherical neighborhood $T(\partial\chi, \mu_0)$. We define Γ as the closure of $T(\partial\chi, \frac{\mu_0}{2})$. As a bounded, closed subset of $T(\partial\chi, \mu_0)$, Γ is compact. Since the Hessian of f_χ is continuous on Γ, it is a bounded function on Γ. We denote by $B > 0$ a bound verifying $\|W_{x_0}\|_{op} \leq B$ for all $x_0 \in \Gamma$, with $\|\cdot\|_{op}$ the operator norm.

Then, for a given surface point $x_0 \in \partial\chi$, we are interested in the sign of $f_\chi(x_0 + \lambda N(x_0))$ for $\lambda > 0$. In this regard, we use the Lagrangian form of the Taylor expansion of f_χ at x_0: For all $\lambda < \frac{\mu_0}{2}$, there exists $w_{x_0,\lambda}$ that lies on the line connecting x_0 and $x_0 + \lambda N(x_0)$ such that

$$f_\chi(x_0 + \lambda N(x_0)) = f_\chi(x_0) + \lambda \nabla f_\chi(x_0)^T N(x_0) + \frac{\lambda^2}{2} N(x_0)^T W_{w_{x_0,\lambda}} N(x_0). \tag{3}$$

Since x_0 is located on the surface, $f(x_0) = 0$. As we mentioned in the previous subsection, $\nabla f_\chi(x_0) = N(x_0)$ so that $\nabla f_\chi(x_0)^T N(x_0) = \|N(x_0)\|_2^2 = 1$.

Moreover, $w_{x_0,\lambda} \in \Gamma$. We apply Cauchy-Schwarz inequality and deduce, by definition of the operator norm:

$$\begin{aligned}
&|N(x_0)^T W_{w_{x_0,\lambda}} N(x_0)| \leq \|N(x_0)\|_2 \cdot \|W_{w_{x_0,\lambda}}\|_{op} \cdot \|N(x_0)\|_2 \\
&|N(x_0)^T W_{w_{x_0,\lambda}} N(x_0)| \leq B.
\end{aligned}$$

Consequently,

$$\begin{aligned}
f_\chi(x_0 + \lambda N(x_0)) &= \lambda + \frac{\lambda^2}{2} N(x_0)^T W_{w_{x_0,\lambda}} N(x_0) \\
&\geq \lambda - \frac{\lambda^2}{2} B.
\end{aligned}$$

The polynomial function $\lambda \mapsto \lambda - \frac{\lambda^2}{2} B$ being positive on $(0, \frac{2}{B})$, we define $\lambda_0 = \min(\frac{2}{B}, \frac{\mu_0}{2})$ and conclude that $f_\chi(x_0 + \lambda N(x_0)) > 0$ for all positive $\lambda < \lambda_0$. □

Finally, we prove the main theorem.

Theorem 1. *Under the previous regularity assumptions on the volume χ of the scene and its surface $\partial\chi$, there exist $\mu_0 > 0$ and $M > 0$ such that for all $\mu < \mu_0$, and any camera $c \in \mathcal{C}$:*

$$\left| \frac{1}{|\chi|_V} \int_\chi g_c^H(\mu; x) dx - \mu \frac{|\partial\chi|_S}{|\chi|_V} G_H(c) \right| \leq M\mu^2 \,, \tag{4}$$

where $|\chi|_V$ is the volume of χ.

Proof. We keep the notations introduced in the previous lemma and, if needed, we update the value of μ_0 so that $\mu_0 \leq \lambda_0$. Then, we start back from equation (3) of the main paper, where we introduced a new visibility gain function g_c^H to adapt the definition of the former visibility gain ν_c^H on spherical neighborhoods. For any $0 < \mu < \mu_0$, we define this function as:

$$g_c^H(\mu; x) = \begin{cases} 1 & \text{if } \exists x_0 \in \partial\chi, \lambda < \mu \text{ such that } x = x_0 + \lambda N(x_0) \text{ and } \nu_c^H(x_0) = 1, \\ 0 & \text{otherwise} \,, \end{cases} \tag{5}$$

where N is the inward normal vector field, which is well defined according to our main regularity assumptions. Once again, for $c \in \mathcal{C}, \mu < \mu_0$, g_c^H is obviously bounded on the bounded subset $T(\partial\chi, \mu_0)$. Thus, $g_c^H(\mu, \cdot)$ is absolutely integrable on $T(\partial\chi, \mu_0)$. Consequently, we apply the formula 2 from [3] and find that, for all $c \in \mathcal{C}$ and $\mu < \mu_0$:

$$\int_{T(\partial\chi,\mu)} g_c^H(\mu; x) \mathrm{d}x = \int_{\partial\chi} \int_{-\mu}^{\mu} g_c^H(\mu; x_0 + \lambda N(x_0)) \, \det(I - \lambda W_{x_0}) \, \mathrm{d}\lambda \, \mathrm{d}x_0. \tag{6}$$

Now, we are going to show that $\det(I - \lambda W_{x_0}) = 1 + \lambda b(\lambda, x_0)$ where b is a bounded function on the compact space $[-\mu_0, \mu_0] \times \partial\chi$. To this end, we could either develop directly the determinant or use results about characteristic polynomials to speed up the proof.

We choose the second approach: since W_{x_0} is a square 3×3 matrix, there exist polynomial functions $f_i : \mathbb{R}^{3\times3} \to \mathbb{R}, i = 1, ..., 3$ such that, for all $x_0 \in \partial\chi$, the characteristic polynomial of W_{x_0} verifies

$$\det(XI_3 - W_{x_0}) = X^3 + f_1(W_{x_0})X^2 + f_2(W_{x_0})X + f_3(W_{x_0}).$$

Consequently, for all $x_0 \in \partial\chi$ and $0 < \lambda < \mu_0$,

$$\begin{aligned}\det(I_3 - \lambda W_{x_0}) &= \lambda^3 \det(\frac{1}{\lambda} I_3 - W_{x_0}) \\ &= \lambda^3 \left(\frac{1}{\lambda^3} + f_1(W_{x_0}) \frac{1}{\lambda^2} + f_2(W_{x_0}) \frac{1}{\lambda} + f_3(W_{x_0}) \right) \\ &= 1 + \lambda \left(f_1(W_{x_0}) + f_2(W_{x_0})\lambda + f_3(W_{x_0})\lambda^2 \right).\end{aligned}$$

We define $b : (\lambda, x_0) \mapsto f_1(W_{x_0}) + f_2(W_{x_0})\lambda + f_3(W_{x_0})\lambda^2$. Since the functions f_i are polynomial and $x_0 \mapsto W_{x_0}$ is continuous on $\partial\chi$, we deduce that b is bounded on $[-\mu_0, \mu_0] \times \partial\chi$ as a continuous function defined on a compact space. We denote by $M > 0$ a bound verifying $|b(\lambda, x_0)| \leq M$ for all $(\lambda, x_0) \in [-\mu_0, \mu_0] \times \partial\chi$.

Subsequently, for all $x_0 \in \partial\chi$, we have by definition $g_c^H(\mu; x_0 + \lambda N(x_0)) = g_c^H(\mu; x_0) = \nu_c(x_0)$ when $0 \leq \lambda < \mu$, and $g_c^H(\mu; x_0 + \lambda N(x_0)) = 0$ when $-\mu < \lambda < 0$. By rewriting equation 6, it follows that, for every $0 < \mu < \mu_0$:

$$\begin{aligned}\int_{T(\partial\chi,\mu)} g_c^H(\mu; x) \mathrm{d}x &= \int_{\partial\chi} \int_0^\mu g_c^H(\mu; x_0)(1 + \lambda b(\lambda, x_0)) \,\mathrm{d}\lambda \,\mathrm{d}x_0 \\ &= \mu \int_{\partial\chi} g_c^H(\mu; x_0) \,\mathrm{d}x_0 + \int_{\partial\chi} \int_0^\mu \lambda g_c^H(\mu; x_0) b(\lambda, x_0) \,\mathrm{d}\lambda \,\mathrm{d}x_0 \qquad (7) \\ &= \mu |\partial\chi|_S G_H(c) + \int_{\partial\chi} \int_0^\mu \lambda g_c^H(\mu; x_0) b(\lambda, x_0) \,\mathrm{d}\lambda \,\mathrm{d}x_0.\end{aligned}$$

Since the volume is supposed to be opaque, function $g_c^H(\mu; \cdot)$ is equal to 0 for every point outside $T(\partial\chi, \mu)$. Moreover, according to lemma 2, for all $x_0 \in \partial\chi, \mu < \mu_0$, the point $x_0 + \mu N(x_0)$ is located inside the volume χ, such that $\int_{T(\partial\chi,\mu)} g_c^H(\mu; x) \,\mathrm{d}x = \int_\chi g_c^H(\mu; x) \,\mathrm{d}x$.

Since $|g_c^H(\mu; \cdot)| \leq 1$ for all $c \in \mathcal{C}$ and $\mu < \mu_0$:

$$\begin{aligned}\left| \int_\chi g_c^H(\mu; x) \mathrm{d}x - \mu |\partial\chi|_S G_H(c) \right| &\leq \int_{\partial\chi} \int_0^\mu \lambda \left| g_c^H(\mu; x_0) b(\lambda, x_0) \right| \,\mathrm{d}\lambda \,\mathrm{d}x_0 \\ &\leq \int_{\partial\chi} \int_0^\mu \lambda \left| b(\lambda, x_0) \right| \,\mathrm{d}\lambda \,\mathrm{d}x_0 \\ &\leq \int_{\partial\chi} \int_0^\mu \lambda M \,\mathrm{d}\lambda \,\mathrm{d}x_0 \qquad (8) \\ &\leq \frac{M\mu^2}{2} \int_{\partial\chi} \mathrm{d}x_0 \\ &\leq \frac{M\mu^2}{2} |\partial\chi|_S = |\chi|_V \cdot M' \mu^2\end{aligned}$$

with $M' = \frac{M}{2} \frac{|\partial\chi|_S}{|\chi|_V}$. □

B Training

Motivated by the literature about NBV reconstruction for objects, we trained our model on the simple case of single, centered object reconstruction (see Next Best View for Single Object Reconstruction), with meshes from ShapeNetCore v1 [1], following the same training, validation and test distributions than [11]. Indeed, we wanted our model to learn general geometric prior and compute coverage gain predictions on various shapes. Since SCONE relies on neighborhood geometric features as well as proxy point-level information to compute various metrics, we made the assumption that training only on an object dataset should not prevent the model from scaling to 3D scenes. We confirmed this hypothesis with an experiment on large 3D structures with free motion on a 5D grid, detailed in Active View Planning in a 3D Scene.

In theory, the full model could be trained directly in an end-to-end fashion with a single loss. However, in practice we schedule the process, and train the two modules consecutively. In particular,

we first train the geometric prediction module alone so that predicted occupancy mappings become meaningful; then, we use the geometric predictions as an input to train the second module, and make it learn to predict accurate visibility gains. Indeed, training the entire model from scratch makes convergence difficult since the second module of the model learns to predict visibility gains from meaningless geometric reconstructions at the beginning.

B.1 Training the geometric prediction module

Inspired by [5, 9], we follow a simple pipeline to train the geometric prediction model: For each mesh M_i in a batch $(M_1, ..., M_{N_{\text{mesh}}})$, we sample a random number n_i of camera poses, capture n_i depth maps from these positions, and compute the corresponding partial point cloud P_i. Then, for each mesh we sample N_X proxy points $X^{(i)} = (x_1^{(i)}, ..., x_{N_X}^{(i)})$ and compute their predicted occupancy probability $\hat{\sigma}(P_i;\ x_j^{(i)})$. Finally, we use an MSE loss to compare the predictions with ground truth occupancy values.

More exactly, the loss $\mathcal{L}_{\text{occ}}$ used to train the geometric prediction module of SCONE is the following:

$$\mathcal{L}_{\text{occ}} = \frac{1}{N_{\text{mesh}}} \sum_{i=1}^{N_{\text{mesh}}} \frac{1}{N_X} \sum_{j=1}^{N_X} \|\hat{\sigma}(P_i;\ x_j^{(i)}) - \sigma(x_j^{(i)})\|^2. \tag{9}$$

B.2 Training the coverage prediction module

Once the geometric prediction model converges, we start to train the visibility gain I_H prediction module. To this end, we first select a batch of meshes. For each mesh, we capture a random number of initial depth map observations (up to 10) with a ray-casting renderer and apply SCONE to predict visibility gain function coordinates in spherical harmonics. Since all cameras are sampled on a sphere in this setup, they share the same proxy points in their field of view and we can directly and efficiently compute coverage gains for a dense set of cameras on the sphere from predicted visibility gains, in a single forward pass.

Since our predicted volumetric gains are not equal to the true coverage gains but only proportional, we cannot compare directly the predicted values to ground-truth coverage gains but have to normalize them first. Moreover, we want SCONE not only to select the right NBVs consistently but also predict an accurate distribution of coverage gains in the volume for further path planning applications. Consequently, we consider the predicted $I_H(c)$ as a distribution on c and compare it to the ground truth coverage gains using the Kullback-Leibler divergence D_{KL} after a softmax normalization.

More exactly, the training process starts in the same way as the previous one: For each mesh M_i in a batch $(M_1, ..., M_{N_{\text{mesh}}})$, we sample a random number n_i of camera poses (which correspond to the history H_i), capture n_i depth maps from these positions, and compute the corresponding partial point cloud P_i. For each mesh we sample N_X proxy points $X^{(i)} = (x_1^{(i)}, ..., x_{N_X}^{(i)})$ and compute their predicted occupancy probability $\hat{\sigma}(P_i;\ x_j^{(i)})$.

Then, for each mesh we sample a subset of proxy points according to their occupancy probability; we concatenate these proxy points with their occupancy values, compute the camera history features $(h_{H_i}(x_1^{(i)}), ..., h_{H_i}(x_{N_X}^{(i)}))$ and feed these inputs to the second module of SCONE to predict the spherical mappings ϕ_l^m of visibility gains with a single forward pass.

Next, we sample a dense set of N_{cam} camera poses $\mathcal{C}^{(i)} = (c_1^{(i)}, ..., c_{N_{\text{cam}}}^{(i)})$ on a sphere around the object and compute the predicted coverage gains $(I_{H_i}(c_1^{(i)}), ..., I_{H_i}(c_{N_{\text{cam}}}^{(i)}))$ for all cameras using Monte-Carlo integration on the predicted spherical mappings of visibility gains. Finally, we compute the following loss $\mathcal{L}_{\text{cov}}$ to supervise training for the second module of SCONE:

$$\begin{aligned} \mathcal{L}_{\text{cov}} &= \frac{1}{N_{\text{mesh}}} \sum_{i=1}^{N_{\text{mesh}}} D_{KL}(\text{softmax}(G_{H_i}) \,||\, \text{softmax}(I_{H_i})) \\ &= \frac{1}{N_{\text{mesh}}} \sum_{i=1}^{N_{\text{mesh}}} \sum_{j=1}^{N_{\text{cam}}} s_j^{(i)} \log\left(\frac{s_j^{(i)}}{\hat{s}_j^{(i)}}\right), \end{aligned} \tag{10}$$

where the softmax normalization is defined as follows:

$$s_j^{(i)} = \frac{\exp\left(G_{H_i}(c_j^{(i)})\right)}{\sum_{k=1}^{N_{\text{cam}}} \exp\left(G_{H_i}(c_k^{(i)})\right)} \quad \text{and} \quad \hat{s}_j^{(i)} = \frac{\exp\left(I_{H_i}(c_j^{(i)})\right)}{\sum_{k=1}^{N_{\text{cam}}} \exp\left(I_{H_i}(c_k^{(i)})\right)}.$$

We supervise on the final coverage value (*i.e.*, the value of the Monte-Carlo integration) rather than per-point visibility gains, since it would be heavier to compute and would require further assumptions. Indeed, we would have to compute ground truth visibility gains for probabilistic points that may not be in χ, which needs further hypothesis to handle properly. On the contrary, supervising on the integral value prevents us from handling directly this problem: it is up to the model to process information as a volumetric integral, identify which point could be in the spherical neighborhood in an implicit manner and learn a meaningful per-point visibility gain metric. We can stick to our volumetric framework and have no need to make further assumptions on geometry or directly extract surfaces from our predicted probability field, which could lead to inaccurate results.

Overall, our model has 3,650,657 parameters: 2,257,769 for the occupancy probability prediction module, and 1,392,888 for the visibility gain prediction module. Each module is trained 20 hours on 4 GPUs Nvidia Tesla V100 SXM2 16 Go. We use a linear warmup strategy on the learning rate λ to make training more stable: λ linearly increases from 0 to its default value $\lambda = 10^{-4}$ during the first 1,000 iterations. The learning rate is ultimately decreased to 10^{-5} after 60,000 iterations for further improvement.

All experiments were run on a single GPU Nvidia GeForce GTX 1080 Ti. Further details about the model's architecture are given in Section Implementation details.

B.3 Implementation details

We implemented and trained our model with PyTorch [6]. In particular, we used ray-casting renderers from PyTorch3D [7] to generate and use depth maps as inputs to our model.

To compute all spherical mappings involved in our method, we use the orthonormal basis of spherical harmonics with rank lower or equal to 7, which makes a total of 64 harmonics. In this regard, both the camera history features h_H and the predicted visibility gain functions ϕ are mapped as vectors with 64 coordinates.

Code and data are available on our dedicated webpage: `https://github.com/Anttwo/SCONE`.

C Experiments

In this section, we give further details about our experiments: how the dataset is constructed, the hyperparameters involved in the prediction, and the evaluation metrics. We also provide additional results.

C.1 Next Best View for Single Object Reconstruction

We first compare the performance of our model to the state of the art on a subset of the ShapeNet dataset [1].

Dataset. We follow the protocol of [11]: Using the train/validation/test split of [10] for ShapeNet dataset [1], we sample 4,000 training meshes from 8 categories of objects (Airplane, Cabinet, Car, Chair, Lamp, Sofa, Table, Vessel), as well as 400 validation meshes and 400 test meshes from the same categories. Note that the full test set used in [10] contains 1200 meshes. To make sure we provide a fair comparison with the state of the art, we sampled 10 different test subsets of 400 meshes among all 1200 meshes, ran the experiment 10 times and averaged the metrics. The results were really close from one subset to another, and our model systematically provided better coverage values than other methods in literature.

Following [11], we reconducted the same experiment with subsets of 400 test meshes from 8 categories unseen during training (Bed, Bench, Bookshelf, Bus, Guitar, Motorbike, Pistol, Skateboard).

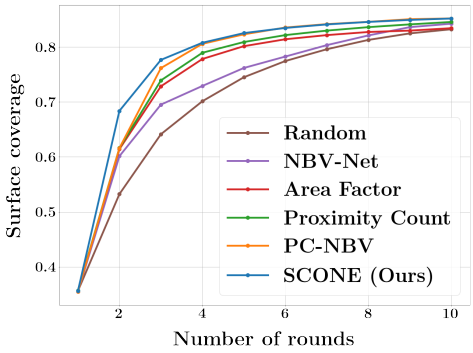


Figure 2: **Convergence speed of the covered surface by several methods (NBV-Net [4], Area Factor [8], Proximity Count [2], PC-NBV [11], ours) on a subset of ShapeNet dataset [1]**. Our SCONE method has the highest reconstruction performance in terms of AUC, and performs significantly better when only a small number of views are available. For this experiment, we constrain the camera to stay on a sphere centered on the objects in order to compare with previous methods.

Prediction. All camera poses are sampled on a sphere around the object. For each mesh, we follow the protocol of [11] and capture a first depth map from a random camera pose, which is reprojected in 3D as a cloud of 1024 points. Then, we use our model to predict coverage gain for all camera poses, and select the NBV as the pose with the highest gain. We concatenate the partial point cloud captured from the new position to the previous one, and we iterate the process until we have 10 views of the object. Note that we reconstruct a small scale object which is entirely contained in the field of view of every camera, *i.e.*, $\chi_c = \chi$ for all camera poses c. Therefore, at each round we compute coverage gains for all camera poses in a single forward pass thanks to the spherical mappings of visibility gain functions.

Evaluation metric. Once again, we follow the protocol of [11] to compute ground-truth coverage scores and evaluate our method. For each mesh, we uniformly sample a cloud P_0 of 16,384 points on the surface, which represent the ground truth surface. Then, the total coverage $C(P_H)$ of a partial point cloud P_H obtained by merging depth maps captured from camera poses in H is computed as follows:

$$C(P_H) = \frac{1}{|P_0|} \sum_{p_0 \in P_0} U(\epsilon - \min_{p \in P_H} \|p_0 - p\|_2), \tag{11}$$

where ϵ is a distance threshold. $C(P_H)$ simply evaluates the number of points in P_0 that have at least a neighbor in P_H closer than ϵ. For the experiment, we used the same threshold as [11], *i.e.*, $\epsilon = 0.00707m$.

Results. We provide additional results for this experiment. In particular, figure 2 illustrates the evolution of surface coverage during reconstruction for several methods.

C.2 Active View Planning in a 3D Scene

Dataset. Since, to the best of our knowledge, we propose the first supervised Deep Learning method for free 6D motion of the camera, we created a dataset made of 13 large-scale scenes under the CC License for quantitative evaluation (3D models courtesy of Brian Trepanier, Andrea Spognetta, and 3D Interiors; all models were downloaded on the website Sketchfab). For each scene, we defined a bounding box delimiting the main structure to reconstruct.

To scale the geometric prediction module of SCONE to large 3D scenes, we partition the scene in 3D cells depending on the dimensions of the bounding box: the larger the bounding box, the more cells in the scene. Then, each point belonging to the partial point cloud P_H gathered by the sensor (computed by reprojecting all depth maps in 3D) is stored in the corresponding cell. To predict the occupancy probability of each proxy point, we only use the points located in neighboring cells. Therefore, the

Table 1: **AUCs of surface coverage in large 3D scenes by SCONE and our two baselines** after averaging over multiple trajectories, with standard deviations. Despite being trained only on centered ShapeNet 3D models, the second module of SCONE is able to generalize to complex scenes and consistently reaches better AUC than the baselines.

	Method		
3D scene	Random Walk	SCONE-Entropy	SCONE
Dunnottar Castle	0.355 ± 0.106	0.456 ± 0.041	**0.739** ± 0.050
Manhattan Bridge	0.405 ± 0.089	0.361 ± 0.065	**0.685** ± 0.034
Alhambra Palace	0.384 ± 0.086	0.437 ± 0.047	**0.567** ± 0.031
Leaning Tower	0.286 ± 0.122	0.415 ± 0.023	**0.542** ± 0.026
Neuschwanstein Castle	0.403 ± 0.032	0.538 ± 0.040	**0.653** ± 0.025
Colosseum	0.308 ± 0.061	0.512 ± 0.024	**0.571** ± 0.024
Eiffel Tower	0.495 ± 0.062	0.741 ± 0.017	**0.762** ± 0.020
Fushimi Castle	0.584 ± 0.078	0.802 ± 0.022	**0.841** ± 0.027
Pantheon	0.175 ± 0.065	0.351 ± 0.020	**0.396** ± 0.036
Bannerman Castle	0.321 ± 0.121	**0.667** ± 0.023	0.642 ± 0.047
Christ the Redeemer	0.600 ± 0.146	0.839 ± 0.038	**0.859** ± 0.022
Statue of Liberty	0.469 ± 0.075	0.681 ± 0.018	**0.693** ± 0.032
Natural History Museum	0.147 ± 0.024	0.080 ± 0.010	**0.177** ± 0.031
Mean	0.380	0.529	**0.625**

growth of the global partial point cloud does not influence the computation time of the occupancy probability prediction: We avoid unnecessary computation but keep meaningful predictions since our geometric prediction model relies on local neighborhood features.

Note that we do not reproject every point of the depth maps to compute the partial point cloud P_H. Indeed, we introduce a distance threshold ϵ to avoid unnecessary large amounts of points in the cells. Let p be a point belonging to a new depth map computed by the sensor; then, we identify the cell in which p is located, and add p to the point cloud P_H if and only if for all points p_0 in the cell, $\|p - p_0\|_2 > \epsilon_{\text{cloud}}$. Therefore, the threshold ϵ_{cloud} is a parameter that influence the resolution of the reconstructed point cloud P_H.

Prediction. To evaluate the scalability of our model to large environments as well as free camera motion in 3D space, we use once again a ray-casting renderer and follow the protocol described in section Active View Planning in a 3D Scene of the main paper. However, to predict surface coverage gain in a large 3D scene, we slightly modify the computation of per-point visibility gains.

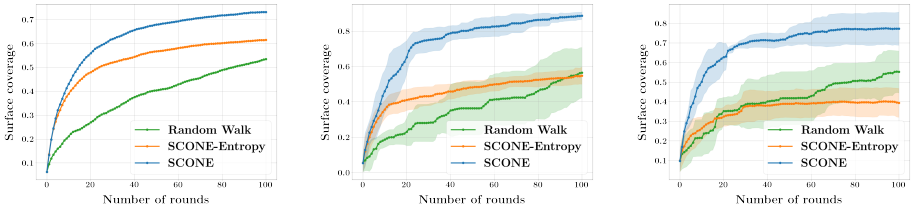
Indeed, note that the density of points gathered by a depth sensor like a LiDAR decreases with the distance to the surface, as well as the angle between the surface normal and the direction of observation. Since SCONE was trained on small scale objects with camera poses sampled on a sphere, our model learned to predict optimized angle for observation, but did not learn to predict optimized distance. Actually, the predicted visibility gain of proxy points sampled in space should reflect the variations in LiDAR density, and decrease inversely with the squared distance to the camera.

To this end, given a camera pose c, we penalize the distance by multiplying the predicted visibility gain of a proxy point $x \in \chi_c$ by a factor $\frac{1}{\eta + \|x - c_{pos}\|_2^2}$. We apply the same strategy to the baseline SCONE-Entropy, and multiply the Shannon Entropy of a proxy point by the same factor.

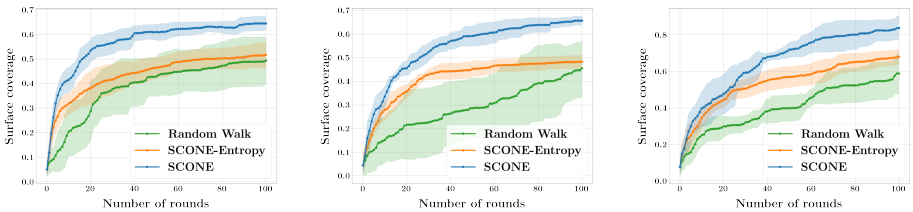
Evaluation metric. To compute ground-truth total surface coverage for evaluation, we use the same approach than the previous experiment detailed in subsection C.1. In particular, we sample 100,000 points on the surface to obtain a ground-truth cloud P_0. Moreover, we use ϵ_{cloud}, the parameter that defines the resolution of the reconstructed point cloud P_H, to compute the total surface coverage following equation 11.

C.3 Ablation study

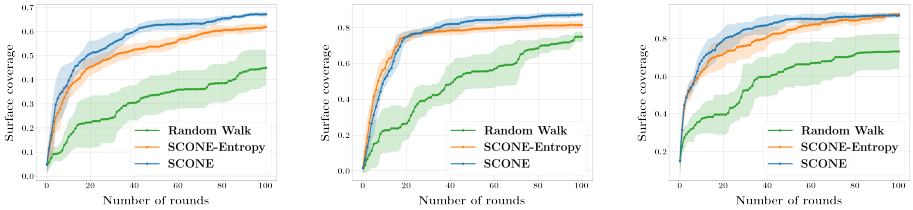
In this subsection, we provide further analysis about both prediction modules of SCONE.



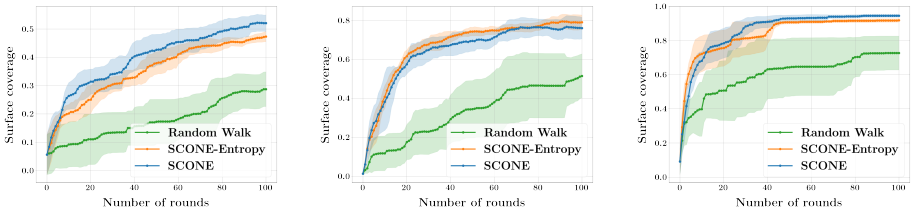
(a) Average on all 13 scenes (b) Dunnottar Castle (c) Manhattan Bridge



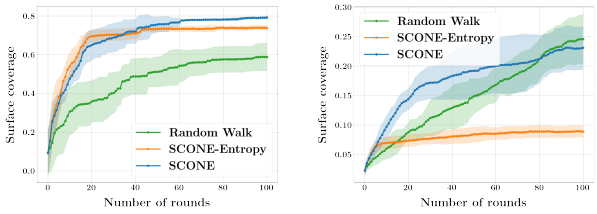
(d) Alhambra Palace (e) Leaning Tower, Pisa (f) Neuschwanstein Castle



(g) Colosseum (h) Eiffel Tower (i) Fushimi Castle



(j) Pantheon (k) Bannerman Castle (l) Christ the Redeemer

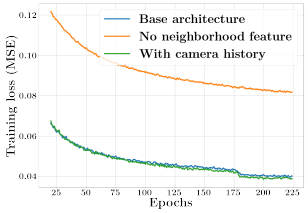


(m) Statue of Liberty (n) Natural History Museum

Figure 3: **Convergence speed of the covered surface in large 3D scenes by SCONE and our two baselines.** The first image shows the average on all scenes. For each scene, surface coverage is averaged on several trajectories starting from different camera poses. Standard deviations are shown on the figures. Despite being trained only on centered ShapeNet 3D models, the second module of SCONE is able to generalize to complex scenes and consistently reaches better coverage than the baselines.

Architecture	Mean Squared Error	Continuous IoU	IoU
Base Architecture	0.0397	0.777	0.843
No Neighborhood Feature	0.0816	0.611	0.702
With Camera History	**0.0386**	**0.782**	**0.844**

(a) MSE and IoU after training



(b) Training loss (MSE)

Figure 4: **(a) Comparison of Mean Squared Error and IoU for variations of our occupancy probability prediction model after training. (b) Comparison of training losses (MSE) for variation of our model.** Without the multi-scale neighborhood features, the occupancy probability prediction module of SCONE suffers from a large decrease in performance. On the contrary, using camera history as an input feature only offers a marginal increase in performance.

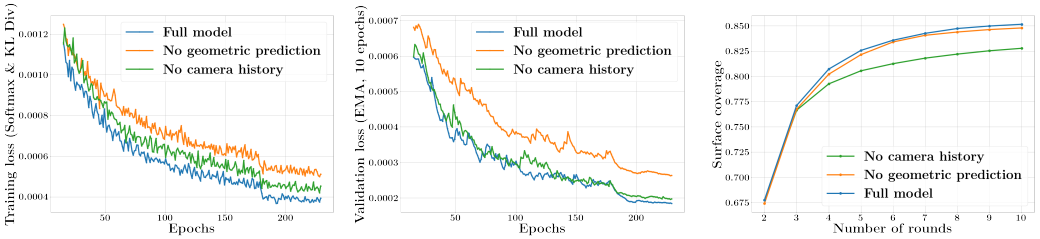
Occupancy probability. As we explained in the main paper, the lack of neighborhood features causes a huge loss in performance. On the contrary, using the spherical mappings $h_H(x)$ of camera history H as an additional feature offers a marginal increase in performance.

In this appendix, we develop our analysis and provide not only the values of the MSE at the end of training but also IoU and training losses that support our conclusions in figure 4. In particular, we compute a Continuous IoU which extends the definition of IoU to a non-binary occupancy probability field. More exactly, we keep notations from subsection B.1, and define the continuous IoU for the i^{th} mesh of the test dataset as

$$\text{IoU}_{\text{continuous}} = \frac{\sum_{j=1}^{N_X} \hat{\sigma}(P_i;\, x_j^{(i)}) \cdot \sigma(x_j^{(i)})}{\sum_{j=1}^{N_X} \hat{\sigma}(P_i;\, x_j^{(i)}) + \sigma(x_j^{(i)}) - \hat{\sigma}(P_i;\, x_j^{(i)}) \cdot \sigma(x_j^{(i)})} \tag{12}$$

We also compute a more conventional IoU by thresholding occupancy probability: We define the set of predicted occupied points as the set of all points with a predicted occupancy probability above 0.5. Then, we compute the IoU with the set of ground truth occupied points.

Visibility gain. We provide in figure 5 additional results supporting the observations we made in the main paper: the geometric prediction computed in a volumetric framework greatly increases performance for computing an accurate distribution of coverage gains for all camera poses in the scene, as the Kullback-Leibler divergence loss suggests. On the contrary, using spherical mappings h_H of camera history as an additional input only offers a marginal increase in performance for computing the distribution of coverage gains. However, camera history features drastically improve the identification of the maximum of the distribution of coverage gains (*i.e.*, the selection of a single NBV), as shown by the evolution of surface coverage during reconstruction.



(a) Training loss (b) Validation loss (c) Surface coverage

Figure 5: **Comparison of (a) training losses, (b) validation losses, and (c) surface coverage for variations of our visibility gain prediction model. The surface coverage is computed during a reconstruction process that follows the protocol presented in section C.1**. The validation loss is plotted with exponentially weighted moving average over 10 epochs. For surface coverage, the first round of reconstruction is not plotted since all curves start from the same point. Thanks to its volumetric approach, the full model predicts a better distribution of coverage gains on the whole space as it is indicated by the KL Divergence training loss, which is convenient for full path planning and trajectory computation in a 3D scene. Moreover, the full model does not suffer from a loss of performance in coverage when selecting a single NBV—*i.e.*, identifying the maximum of the coverage gain distribution—compared to the version that uses directly the dense surface points, which is generally the case when working with volumetric approaches.

References

[1] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical report, Stanford University, Princeton University, Toyota Technological Institute at Chicago, 2015.

[2] Jeffrey Delmerico, Stefan Isler, Reza Sabzevari, and Davide Scaramuzza. A Comparison of Volumetric Information Gain Metrics for Active 3D Object Reconstruction. *Autonomous Robots*, 2018.

[3] David Gilbarg and Neil S. Trudinger. *Elliptic Partial Differential Equations of Second Order*. Classics in Mathematics. Springer Berlin Heidelberg, 2001.

[4] Miguel Mendoza, Juan Irving Vasquez-Gomez, Hind Taud, Luis Enrique Sucar, and Carolina Reta. Supervised Learning of the Next-Best-View for 3D Object Reconstruction. *Computing Research Repository*, 2019.

[5] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. *International Conference on Computer Vision and Pattern Recognition*, 2019.

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, 2019.

[7] Nikhila Ravi, Jeremy Reizenstein, David Novotný, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *CoRR*, 2020.

[8] Juan Vasquez-Gomez, Luis Sucar, Rafael Murrieta-Cid, and Efrain Lopez-Damian. Volumetric Next-Best-View Planning for 3D Object Reconstruction with Positioning Error. *International Journal of Advanced Robotic Systems*, 2014.

[9] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomír Mech, and Ulrich Neumann. DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction. *Advances in Neural Information Processing Systems*, 2019.

[10] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: Point Completion Network. In *International Conference on 3D Vision*, 2018.

[11] Rui Zeng, Wang Zhao, and Yong-Jin Liu. PC-NBV: A Point Cloud Based Deep Network for Efficient Next Best View Planning. In *International Conference on Intelligent Robots and Systems*, 2020.