Progressive integration of visibility constraints for implicit functions

Simen Haugo, Annette Stahl

Simen Haugo, Annette Stahl, "Progressive integration of visibility constraints for implicit functions," Proc. SPIE 12084, Fourteenth International Conference on Machine Vision (ICMV 2021), 1208405 (4 March 2022); doi: 10.1117/12.2622458

Event: Fourteenth International Conference on Machine Vision (ICMV 2021), 2021, Rome, Italy

Progressive Integration of Visibility Constraints for Implicit Functions

Simen Haugo and Annette Stahl

Department of Engineering Cybernetics
Norwegian University of Science and Technology
Trondheim, Norway

ABSTRACT

Procedurally-defined implicit functions, such as CSG trees and recent neural shape representations, offer compelling benefits for modeling scenes and objects, including infinite resolution, differentiability and trivial deformation, at a low memory footprint. The common approach to fit such models to measurements is to solve an optimization problem involving the function evaluated at points in space. However, the computational cost of evaluating the function makes it challenging to use visibility information from range sensors and 3D reconstruction systems. We propose a method that uses visibility information, where the number of function evaluations required at each iteration is proportional to the scene area. Our method builds on recent results for bounded Euclidean distance functions by introducing a coarse-to-fine mechanism to avoid the requirement for correct bounds. This makes our method applicable to a greater variety of implicit modeling techniques, for which deriving the Euclidean distance function or appropriate bounds is difficult.

Keywords: 3D registration, implicit modeling, visibility constraints

1. INTRODUCTION

An implicit surface represents a geometric object as a function $f_x : \mathbb{R}^3 \times \mathcal{X} \to \mathbb{R}$, where the object's interior, exterior and surface are defined implicitly by the function's value in space. Procedurally-defined functions, in particular, enable a powerful framework of constructive modeling techniques [1–5], and have recently led to breakthroughs in shape learning [6, 7]. For computer vision purposes, the function may be parameterized by shape-controlling parameters $x \in \mathcal{X}$ to span a space of possible shapes, which can be searched for a specific instance that best fits incoming measurements. However, the lack of an explicit surface and the computational cost of evaluating the function makes this search challenging. Specifically, in the absence of an explicit surface, a commonly used approach for determining the best fitting parameters is to solve an optimization problem of the form

$$x^* = \arg\min_{x \in \mathcal{X}} E(x), \tag{1}$$

where E is an objective function that involves the function f_x evaluated at points in space [8–12]. Depending on the domain $\mathcal{X}$, solving (1) may require substantial sampling of the search space [9–12]. Thus, when f_x is computationally costly to evaluate, the number of evaluations becomes a bottleneck affecting the choice of E. A common choice, given a set of surface point measurements $p \in \mathbb{R}^3$, is the sum of squared function values at each point:

$$E(x) = \sum_p f_x^2(p). \tag{2}$$

When f_x is a distance function to the surface, the optimization of (2) is a point-to-implicit Iterative Closest Point (ICP) method [13–15]. Such methods are efficient in the sense that the required number of function evaluations is proportional to the scene's surface area. However, it is known that the model can be ill-constrained by surface measurements alone when the surface is only partially visible, e.g. due to capturing data from an inadequate set of viewpoints, occlusions, sensor noise or specular reflections [8]. Regularization [18], such as penalizing description length, penalizing volume or surface area, or imposing parameter constraints can alleviate the issue [8, 11, 19], but are model-specific strategies that do not generalize.

Send correspondence to: {simen.haugo,annette.stahl}@ntnu.no

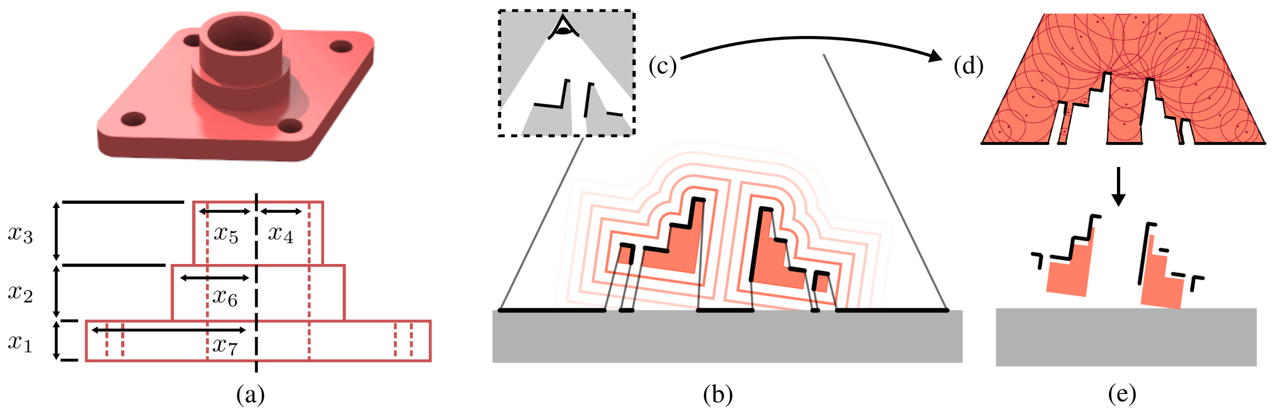

Figure 1. We are given a geometric model (a) as a parameterized implicit function and seek to fit it to incomplete range sensor data (b). To resolve ill-constrained parameters (*e.g.* x_1 and x_7), previous work [16, 17] use visibility information (c) to derive a set of free space consistency constraints (d). For general implicit functions, this derivation relies on bounding the associated Euclidean distance function to the model's surface. Incorrect bounds can lead to a poor fit (e).

Alternatively, one could make use of visibility information (Fig. 1 (c)), that is often provided by sensors and 3D reconstruction systems. For example, a finite range value from a laser scanner indicates not only that there is a surface at that distance from the sensor, but also that there are no surfaces in-between. This imposes additional *free space constraints*; besides fitting the surface measurements, the model must also not occupy space observed to be empty. These constraints have been utilized for fitting both implicit and explicit representations in various ways, such as volume matching [12,20,21], minimizing reprojection error [22–25] or matching of occluding contours or silhouettes [26–28]. For general implicit models, these methods require dense sampling of f_x in the volume or along viewing rays, which can be prohibitively expensive. More efficient methods have recently been proposed for the special case when f_x is either exactly the signed Euclidean distance function to the surface, or when f_x can be bounded from above and below by known functions of the Euclidean distance [16, 17]. However, there is no closed-form expression for the Euclidean distance to several implicit modeling techniques of interest, including set operations, blending and non-isometric deformations [1, 3]. Deriving the necessary upper and lower bounds can also be challenging [17] and the use of incorrect bounds can lead to an inaccurate fit (Fig. 1 (e)).

Contributions. We propose a fitting method for implicit functions that efficiently integrates visibility information. Like previous work [17], our method requires only that an approximation of the Euclidean distance to the model surface can be computed at a given point. However, our method is robust against errors in this approximation. Specifically, we formulate a sequence of problems in which free space constraints are progressively satisfied. In the limit of this sequence, these constraints become non-negativity constraints, which do not depend on the magnitude of the Euclidean distance function. Hence, the influence of any approximation error diminishes to zero. This relaxes the need for correct bounds and makes our method more easily applicable to practical implicit modeling techniques. Additionally, the previous works [16, 17] are efficient in the sense that they compute and utilize the smallest set of function evaluation points to test if the free space constraints are satisfied (i.e. necessary and sufficient conditions). We prove that our method has a similar theoretical complexity in terms of the number of function evaluations, and provide experimental results suggesting that our method in practice may require fewer function evaluations.

2. METHOD

Our work builds on results from [16, 17]. We therefore summarize their relevant findings in Section 2.1 while introducing necessary notation. We describe our method in the subsequent sections.

2.1 Notation and problem formulation

Let $\mathcal{S}$ be a solid representing the scene and let $\mathcal{V}$ be a domain of interest containing $\mathcal{S}$. Free space $\mathcal{V}_{\text{free}}$ is a closed subset of $\mathcal{V}$ that is observed to be empty. The free space boundary $\partial\mathcal{V}_{\text{free}}$ is the boundary between free and unobserved space. The visible surface $\mathcal{R}$ is the observed subset of the physical scene surface $\partial\mathcal{S}$. We denote points in $\mathbb{R}^3$ by c, p, and q. The signed Euclidean distance to a solid $\mathcal{D}$ is denoted $d_{\mathcal{D}}(p) := \pm \min_{q\in\partial\mathcal{D}} ||p-q||_2$, where the sign is negative for p inside $\mathcal{D}$ and positive outside. A ball with center c and radius r is denoted (c, r).

We assume that a geometric model is given as a function $f_x(p) : \mathbb{R}^3 \to \mathbb{R}$. Our goal is to estimate the parameters $x \in \mathcal{X}$ such that the resulting solid $\mathcal{M}_x = \{p \in \mathbb{R}^3 : f_x(p) \leq 0\}$ is consistent with the observed surfaces and free space (*e.g.* the thick lines and un-shaded region in Fig. 1 (b) and (c) respectively). Formally:

$$f_x(p) = 0, \forall p \in \mathcal{R} \text{ and} \tag{3}$$
$$f_x(p) > 0, \forall p \in \text{int } \mathcal{V}_{\text{free}}. \tag{4}$$

It has been shown that these conditions can be turned into an optimization problem of the form [16]

$$\min_x \qquad E(x) = \sum_{p\in\mathcal{R}} f_x(p)^2 \tag{5}$$
$$\text{subject to} \qquad f_x(c) \geq r(c) \quad \forall c \in \mathcal{I}, \tag{6}$$

where $\mathcal{I}$ is a set of points in $\mathcal{V}_{\text{free}}$ and $r(c)$ is the radius of the largest ball at c that is empty with respect to $\mathcal{V}_{\text{free}}$ (*i.e.* its interior contains no point of $\partial\mathcal{V}_{\text{free}}$). When the model is the Euclidean distance function ($f_x = d_{\mathcal{M}_x}$), it has been shown that letting $\mathcal{I}$ be the medial axis of $\mathcal{V}_{\text{free}}$ gives a necessary and sufficient set of free space constraints (6), to ensure that a feasible solution is consistent with free space [16].

A generalization of the above optimization problem has also been proposed for bounded distance functions satisfying [17]

$$g(d_{\mathcal{M}_x}(p)) \leq \lambda^{-1} f_x(p) \leq d_{\mathcal{M}_x}(p), \quad \forall (c, x) \in (\mathbb{R}^3, \mathcal{X}) : f_x(c) > 0, \tag{7}$$

where $g : \mathbb{R} \to \mathbb{R}$ is a non-decreasing function ($g(a) \geq g(b)$ for all $a > b$) and $\lambda > 0$ is a Lipschitz constant, which have both been derived for various implicit modeling techniques and primitives [3, 17]. This leads to a corrected set of free space constraints of the form

$$\lambda^{-1} f_x(c) \geq g(r(c)), \tag{8}$$

which shrinks each ball $(c, r(c))$ to have a radius equal to the lower bound $g(r(c))$, as illustrated in Fig. 2. If the constraint (8) is satisfied, $\mathcal{M}_x$ is guaranteed to be outside the shrunk ball $(c, g(r(c)))$. If the constraint is violated, a subset of $\mathcal{M}_x$ is contained by the original ball $(c, r(c))$, though not necessarily by the shrunk ball. A necessary and sufficient set of constraints (*i.e.* the set of points $\mathcal{I}$) has also been derived for the corrected constraints [17].

2.2 Progressive formulation

To support the case when f_x is not the Euclidean distance function, the above method requires the determination of a function g for the lower bound and a Lipschitz constant λ for the upper bound. If either g or λ is incorrect, i.e. (7) does not hold everywhere, then the resulting constraints can lead to a poor fit, as shown in Fig. 1 (e) compared with Fig. 1 (b). As a conservative estimate, one may choose a large Lipschitz constant and a lower bound close to zero. However, this increases the number of free space constraints and is limited by computational cost. We therefore seek a reformulation of the above method which is robust against incorrect bounds. Our method still assumes that an upper and lower bound is given in the form of a function g and a Lipschitz constant λ, but we ensure that any error in g and λ that causes (7) to not hold will not affect the final model fit.

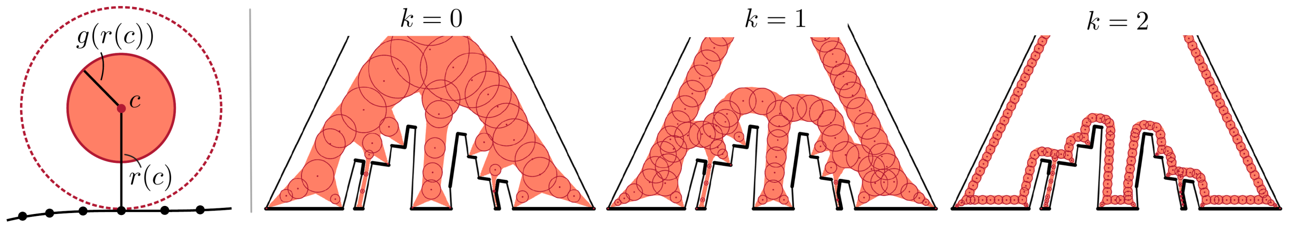

Figure 2. **Left**: Correcting a constraint at c involves shrinking the ball $(c, r(c))$ to accommodate the largest possible error between f_x and $d_{\mathcal{M}_x}$. The shaded region indicates the subset of free space that is "covered" by the new constraint, *i.e.* guaranteed to not contain the model solid $\mathcal{M}_x$ when satisfied. **Right**: Example of a sequence of optimization problems for the scene in Fig. 1.

2.2.1 Sequence of optimization problems

Our method is based on progressively covering free space, rather than all at once. We achieve this by solving a sequence of constrained optimization problems, $k = 0, 1, ...$, in which the k'th *problem instance* is of the form

$$\min_{x} \qquad E(x) = \sum_{p \in \mathcal{R}} f_x(p)^2 \tag{9}$$

$$\text{subject to} \qquad \lambda^{-1} f_x(c_{p,k}) \geq g(r(c_{p,k})) \quad \forall p \in \mathcal{P}_k, \tag{10}$$

where

$$c_{p,k} = p + t_{p,k} n_p \tag{11}$$

is an extension of point p along its associated normal vector n_p by the amount $t_{p,k}$. The set $\mathcal{P}_k \subseteq \partial\mathcal{V}_{\text{free}}$ is a subset of points on the free space boundary. A straightforward choice is $\mathcal{P}_k = \partial\mathcal{V}_{\text{free}}, \forall k$. However, we discuss the issues of this and propose an improvement in Section 2.2.3. The extension amounts $t_{p,k}$ are initialized to $t_{p,0} = r(c_p)$, such that the balls are initially on the medial axis. They are decreased per instance k according to a *reduction schedule*, which we describe in Section 2.2.2.

Fig. 2 shows an example problem sequence for the scene in Fig. 1, using a correct g and λ. The shaded region is the subset of free space covered by the inequality constraints in the k'th problem instance. When satisfied, $\mathcal{M}_x$ is guaranteed to be outside the union of balls $\{c_{p,k}, g(r(c_{p,k}))\}_{p \in \mathcal{P}_k}$. A subset of these balls are drawn with their outline and center for illustration. In the limit, as $k \to \infty$, the inequality constraints become non-negativity constraints ($\lambda^{-1} f_x(p) \geq 0$) on the free space boundary. These only depend on the sign of f_x and not its magnitude. Consequently, the influence of an error in g or λ diminishes as k progresses.

2.2.2 Choosing the reduction schedule

The reduction schedule should ensure that each point in free space is covered by an inequality constraint from at least one problem instance in the sequence. To achieve this, we use the following result [17]. Consider the line segment $[0, r(c_p)]$ connecting a point $p \in \partial\mathcal{V}_{\text{free}}$ to the point c_p on the medial axis along the boundary normal n_p. A necessary and sufficient set $\mathcal{I}$, that fully covers free space, can be obtained by generating additional balls on and along each line segment, such that each line segment is fully contained and none of the balls on a particular line segment overlap. This is achieved by solving

$$\min_{l_{p,i}} \quad l_{p,i} \quad \text{subject to} \quad l_{p,i} + g(r(c_{p,i})) \geq l_{p,i-1} - g(r(c_{p,i-1})) \tag{12}$$

successively for $l_{p,i}$ starting with $l_{p,0} = r(c_p)$. Solving (12) for $l_{p,i}$ gives the center $c_{p,i} = p + l_{p,i} n_p$ of the i'th ball, which has radius $g(r(c_{p,i}))$, on the line segment $[0, r(c_p)]$. We can use this result to obtain a reduction schedule by letting the extension amounts $t_{p,k}$ in the k'th problem instance be the solutions of (12), where $i = k$ and $l_{p,i-1} = t_{p,k-1}$. Depending on g, this may produce an infinite sequence of problem instances, which may be truncated in different ways. For example, by terminating once the remaining distance ($\max_p t_{p,k}$) is less than a given threshold, or by setting $t_{p,k}$ to its limit value (0) when below a given threshold.

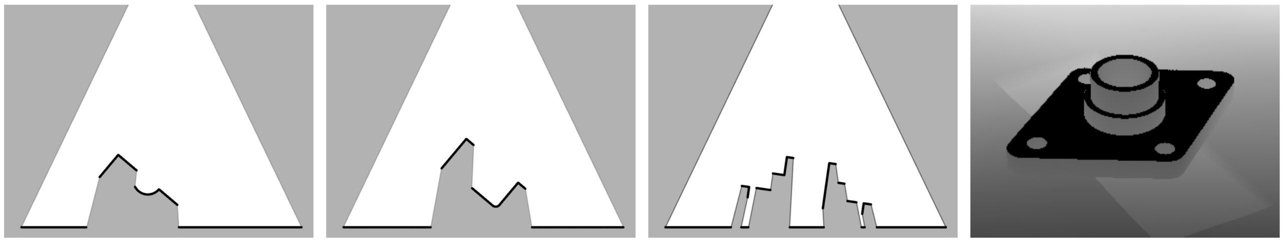

Figure 3. Scenes used in the experiments. In Scene A-C, the colors indicate observed free space (white), unobserved space (gray) and the visible surface (black lines). In Scene D, the colors indicate the measured depth from the range sensor, with the completely black regions being missing ("corrupt") measurements.

2.2.3 Approximate cover and choosing $\mathcal{P}_k$

In a straightforward implementation, one can let $\mathcal{P}_k = \mathcal{V}_{\text{free}}$ in each problem instance k. However, this can be unnecessarily computationally expensive, as the inequality constraints can overlap and therefore be redundant. The amount of redundancy is greater in the earlier instances when the balls are closer to the medial axis. In [16], it was proposed to compute an *approximate cover* of free space—a simplified set of inequality constraints that cover an approximation of free space using fewer constraints. We can likewise compute an approximation of the covered region in each problem instance k. We apply a similar heuristic as [16], where redundant balls are identified by checking if a given ball can be completely covered by enlargening any one of its neighboring balls by adding an amount δ to their radius. The amount δ controls the degree of simplification. Using the reduction schedule from Section 2.2.2, the approximate cover for each instance k can be pre-computed, as the ball locations and radii are determined from the input data alone.

3. EXPERIMENTS

In order to evaluate the proposed approach, we compare our "progressive method" against the method of Haugo and Stahl [17] for bounded distance functions, which we refer to as the "full cover method" in the following. In particular, we investigate the proposed robustness and show that the full cover method has an unstable global solution when (7) does not hold (i.e. the upper and lower bounds are incorrect), whereas our progressive method still has a stable global solution. We also compare the computational cost of the two methods.

3.1 Experiment protocol

3.1.1 Data sets

We consider scenes captured by a range image sensor (Figure 3). Each scene has an associated implicit function model f_x. Scenes A-D are synthesized by simulating a range sensor observing a model from a single viewpoint, for a set of model parameters x^* (ground-truth). In these scenes, the ground-truth parameters can only be inferred when free space constraints are included, and cannot be inferred from surface measurements alone. The models are created using constructive solid geometry and analytical distance bounds to various geometric primitives [3]. We implement set operations using the $\min/\max$ operators, which have been shown [3] to preserve the upper bound $|f_x| \leq |d_{\mathcal{M}_x}|$, such that $\lambda = 1$. However, f_x can still underestimate $d_{\mathcal{M}_x}$. A lower bound g for solids produced by $\min/\max$ operators, is of the form [17]

$$g(d) = \alpha d, \tag{13}$$

where $\alpha \in [0, 1]$ is a constant, which we determined by sampling the value of $f_x(p)/d_{\mathcal{M}_x}(p)$ at the ground-truth parameters x^* and with p sampled regularly in a box enclosing the model solid. For Model A-D this gave a value α of 0.45, 0.09, 0.7 and 0.45, respectively.

3.1.2 Implementation details

Pre-processing pipeline. Both methods require a sampling of the visible surface $\mathcal{R}$ and the set of free space constraints $\mathcal{I}$ (full) or $\mathcal{P}_k$ (progressive). For range images we can backproject the depth pixels into the scene, resulting in a dense point sampling of $\mathcal{R}$. In our scenes, some points belong to unmodeled structures, such as the ground plane. We identify and skip

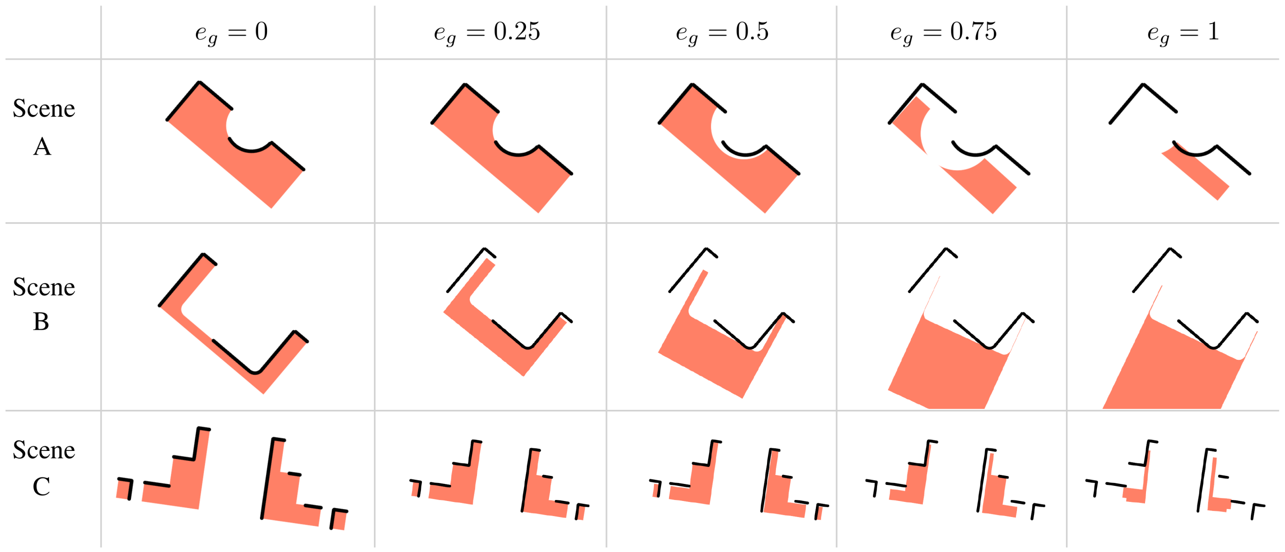

Figure 4. Global solution obtained with the full cover method for different bound errors (increasing left to right).

over these points in the objective functions (5) and (9). The free space constraints can be computed from the medial axis of $\mathcal{V}_{\text{free}}$, if this is represented by a set of medial ball centers c_p, each lying along the normal of a point $p \in \partial\mathcal{V}_{\text{free}}$. The method of Ma *et al.* [29] produces this representation, given an oriented point sampling of the free space boundary $\partial\mathcal{V}_{\text{free}}$. We obtain this oriented point sampling by connecting the adjacent backprojected pixels into a piecewise linear surface, then sampling evenly-distributed points from the surface, and finally estimating their normals by local plane fitting.

The free space constraints for the full cover ($\mathcal{I}$) are then obtained by collecting the sequence of balls produced by solving (12), for each line segment pc_p between the free space boundary and the medial axis. We truncate infinite sequences when the i'th ball is within a tolerance τ of the surface, *i.e.* $t_{p,i} < \tau$. We set the final ball in the sequence to the limit ball ($t_{p,i} = 0$ and $r(c_{p,i}) = 0$). Of these, we exclude balls of zero radius that lie on the visible surface $\mathcal{R}$, as these do not contribute any information. We simplify the constraints by computing an approximate cover using the heuristic described in Section 2.2.3, with δ equal to the inverse of the sampling density of $\partial\mathcal{V}_{\text{free}}$. To compute $\mathcal{P}_k$, we assume that the reduction schedule described in Section 2.2.2 is used. Then, $\mathcal{P}_k$ is simply the k'th solution of (12), combined for all segments pc_p. We likewise truncate the sequence as described above.

Optimization problem solver. We reformulate each constrained problem into the unconstrained problem

$$\min_x \sum_{p\in\mathcal{R}} f_x(p)^2 + \mu \sum_{c\in\mathcal{C}} [\min(f_x(c) - g(r(c)), 0)]^2 \,, \tag{14}$$

where $\mathcal{C} = \mathcal{I}$ for the full cover method and $\mathcal{C} = \mathcal{P}_k$ for our method. The penalty factor μ is initialized to a value μ_0 and increased during optimization. To solve the unconstrained problem, we use a derivative-free solver. When the change in x in two consecutive iterations is less than a threshold, $||x - x_{\text{prev}}||_2 < \tau_x$, we increase μ by a factor of 10 until μ reaches a maximum value of 10^8. We terminate the optimization when $||x - x_{\text{prev}}||_2 < \tau_x$ and $\mu = 10^8$. For our method, rather than solving a new unconstrained problem at each instance k, we apply a single step of the reduction schedule when $||x - x_{\text{prev}}||_2 < \tau_x$. This is done together with increasing μ_0. We terminate our method when $||x - x_{\text{prev}}||_2 < \tau_x$, $\mu_0 = 10^8$ and all free space constraints have been at their limit value ($\max_p t_{p,k} = 0$).

3.1.3 Evaluation variables and metrics

Normalized lower bound error (e_g). By construction, the models satisfy the upper bound $|f_x| \leq |d_{\mathcal{M}_x}|$. The lower bound g is determined by the constant α. To evaluate how the two methods compare with incorrect bounds, we study their

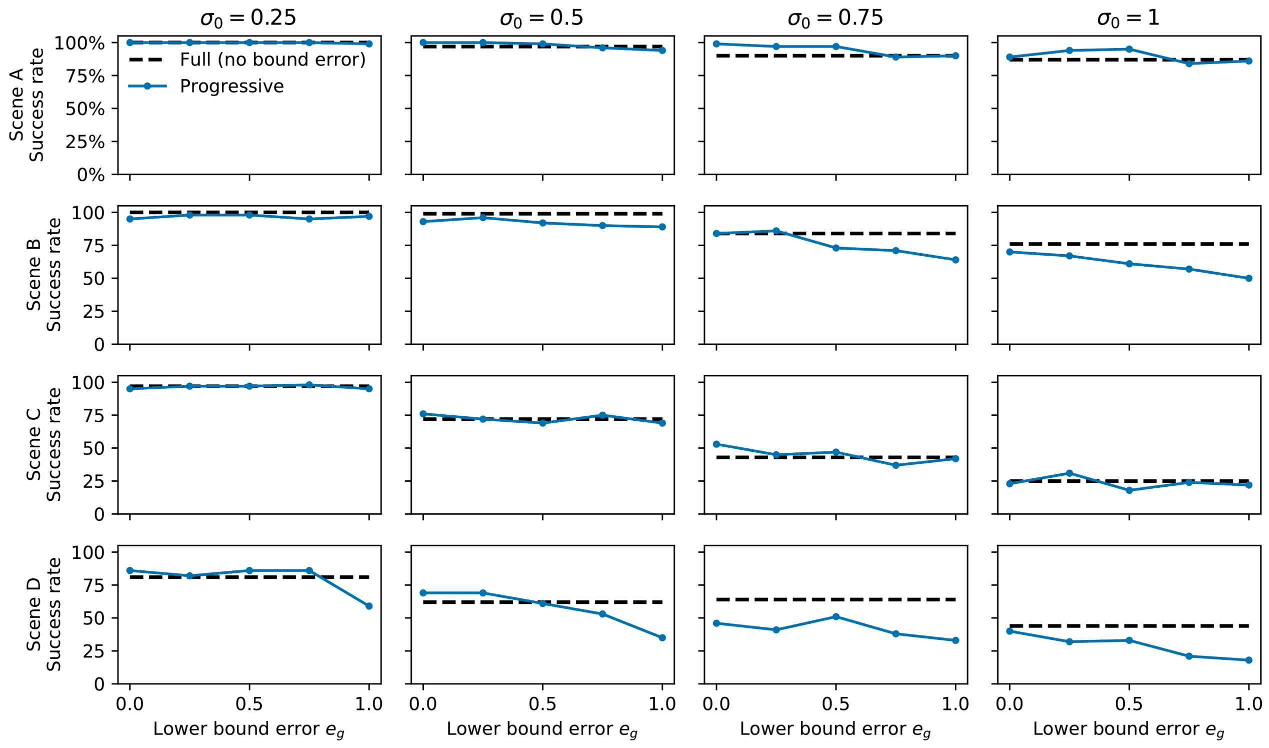

Figure 5. Success rate of our progressive method as a function of lower bound error e_g for each scene and initialization uncertainty. For comparison, we include the success rate of the full cover method at $e_g = 0$ (no bound error).

performance when the constant α varies between its correct value (described in Section 3.1.1 and hereon denoted α^*) and the maximum value $\alpha = 1$. We define the *normalized lower bound error* $e_g \in [0, 1]$ and let

$$\alpha = (1 - e_g)\alpha^* + e_g \ . \tag{15}$$

Thus, when $e_g = 0$, the lower bound is correct. When $e_g = 1$, the lower bound is assumed to be equal to $d_{\mathcal{M}_x}$.

Initialization uncertainty (σ_0)**.** To assess the stability of our results to different starting points and optimization trajectories, we repeat our experiments using different initializations of x. In each run, we initialize x by uniformly sampling from a hypercube of width σ_0 centered around the ground-truth parameters x^*.

Success rate. Each scene has a ground-truth solution x^*. We define success rate as the number of times that the acquired solution is sufficiently close to ground-truth, as measured by $||x - x^*||_2 < \tau_s$, divided by the total number of trials. We set the success threshold τ_s by inspecting the distribution of solutions based on $||x - x^*||_2$ and identifying a threshold below which solutions are indistinguishable from ground-truth. Because the optimization behavior depends on the choice of μ_0, we try $\mu_0 \in \{0.01, 0.1, 1, 10\}$ in each experiment and report the highest success rate.

3.2 Stability in the presence of incorrect bounds

First, we demonstrate the instability of the full cover method in the presence of incorrect bounds. We consider five different values of the normalized lower bound error e_g between 0 and 1. Fig. 4 shows the global solution for the different scenes and bound errors. In the absence of bound error ($e_g = 0$), the solution is identical to ground-truth and consistent with surface measurements and free space. However, the solution is unstable since it diverges from ground-truth as the bound error increases.

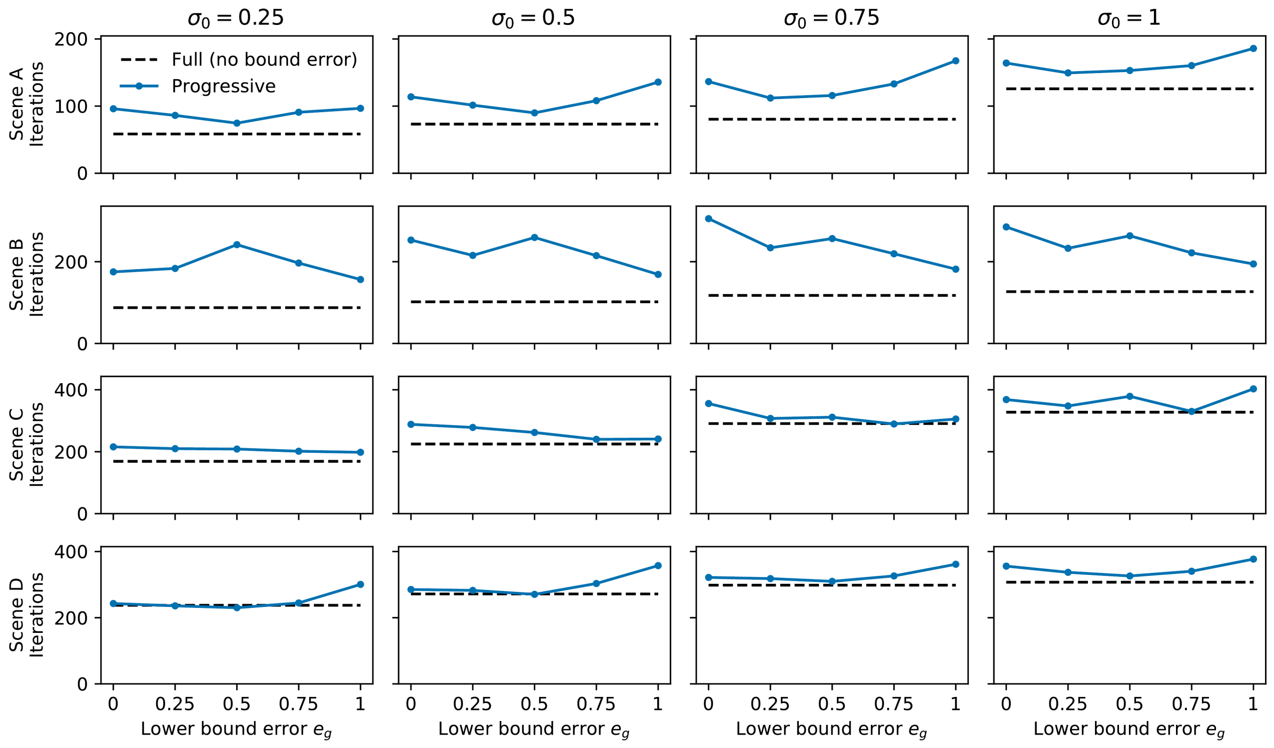

Figure 6. Number of iterations until convergence for successful runs.

Next, we study the stability of our method, using the same bound error magnitudes as above. We perform 100 runs with different initializations, for multiple initialization uncertainty magnitudes ($\sigma_0 \in \{0.25, 0.5, 0.75, 1\}$), and compute the success rate. As a baseline comparison, we include the full cover method's success rate at the given initialization uncertainty and at zero bound error (its success rate would be zero for all non-zero bound errors due to diverging). Fig. 5 shows that our method converges to the ground-truth solution (*i.e.* the left-most column in Fig. 4) even in the presence of incorrect bounds. The success rate is often similar to the baseline, although a significant drop is observed in Scene B and D as the bound error increases. We found in these scenes that the reason for failure was typically by falling into a local optimum early in the optimization, in the attempt to satisfy grossly incorrect free space constraints. These results indicate that our method has a stable global solution in the presence of incorrect bounds, and also that this solution may, depending on the scene and model, be obtained at a similar frequency as the full cover method with correct bounds.

3.3 Computational efficiency

As our method does not involve all free space constraints in each optimization iteration, we expect that our method will require more iterations in total to converge, compared with the baseline (full cover method at zero bound error). Fig. 6 shows that this was the case in almost every experiment. Despite this, Fig. 7 shows that our method used fewer function evaluations. We may compare the expected computational complexity of both methods at zero bound error, by considering the required number of evaluations N_f of the model function f_x, which is typically the dominating cost. Here we will assume that the approximate cover is disabled for both methods. Consider the full cover method and let N_{iter} be the number of optimization iterations until convergence. The resulting number of function evaluations is then

$$N_f = N_{\text{iter}} \cdot (|\mathcal{I}| + |\mathcal{R}|). \tag{16}$$

The size of $\mathcal{I}$ is determined by the length of the ball sequences generated by (12). Let L_p be the length of the ball sequence for point $p \in \partial\mathcal{V}_{\text{free}}$. In the worst case, each ball sequence has length equal to that of the longest sequence $L_{\max} = \max_p L_p$. The total number of function evaluations in the worst case is therefore

$$N_f = N_{\text{iter}} \cdot (L_{\max} \cdot |\partial\mathcal{V}_{\text{free}}| + |\mathcal{R}|). \tag{17}$$

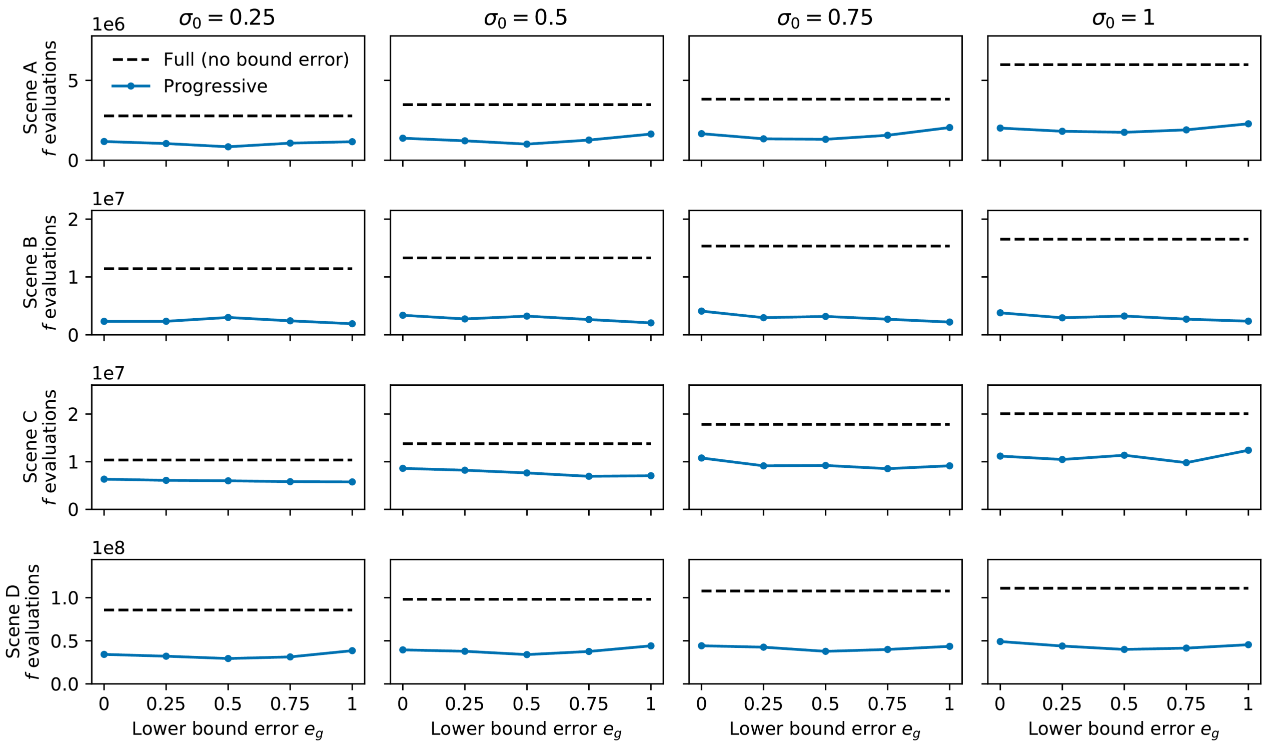

Figure 7. Number of evaluations of the model function f_x until convergence for successful runs.

In our method, the number of problem instances is equal to the maximum ball sequence length L_{max}. Each optimization iteration in the k'th problem instance involves $|\mathcal{P}_k| + |\mathcal{R}|$ function evaluations. Without the approximate cover, $|\mathcal{P}_k| = |\partial\mathcal{V}_{\text{free}}|$. Hence, the total number of function evaluations, assuming each instance requires N_{iter} iterations to converge, is

$$N_f = L_{\text{max}} \cdot N_{\text{iter}} \cdot (|\partial\mathcal{V}_{\text{free}}| + |\mathcal{R}|). \tag{18}$$

Based on this analysis, our method therefore requires an additional $(L_{\text{max}} - 1) \cdot N_{\text{iter}} \cdot |\mathcal{R}|$ function evaluations compared with the full cover method. However, in our experiments we found that our method did not require N_{iter} in each problem instance $k = 0...N_k$, which explains why our method used fewer iterations than expected by this worst case analysis.

4. CONCLUSION

We have presented a method for efficiently integrating visibility information in implicit surface fitting. Our method requires only that an upper and lower bound of the distance to the model surface can be computed at a given point. By satisfying free space constraints progressively rather than all simultaneously, our method is robust against incorrect bounds. This enables a rich class of implicit representations and modeling techniques to more easily be used in applications involving fitting to incomplete data, e.g. in 3D reconstructions of scenes viewed from a limited set of viewpoints or in 3D scans of objects with specular surfaces.

ACKNOWLEDGMENTS

This work is partly supported by the Research Council of Norway through the Centre of Excellence funding scheme, project number 223254, NTNU AMOS.

REFERENCES

[1] Pasko, A., Adzhiev, V., Sourin, A., and Savchenko, V., "Function representation in geometric modeling: concepts, implementation and applications," *The Visual Computer* **11** (1995).

[2] Barr, A. H., "Global and local deformations of solid primitives," in [*SIGGRAPH*], ACM (1984).

[3] Hart, J. C., "Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces," *The Visual Computer* **12** (1996).

[4] Seyb, D., Jacobson, A., Nowrouzezahrai, D., and Jarosz, W., "Non-linear sphere tracing for rendering deformed signed distance fields," *ACM Trans. Graph.* **38** (2019).

[5] Savchenko, V. and Pasko, A., "Transformation of functionally defined shapes by extended space mappings," *The Visual Computer* **14** (1998).

[6] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S., "Deepsdf: Learning continuous signed distance functions for shape representation," in [*CVPR*], IEEE (2019).

[7] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A., "Occupancy networks: Learning 3D reconstruction in function space," in [*CVPR*], IEEE (2019).

[8] Solina, F. and Bajcsy, R., "Recovery of parametric models from range images: The case for superquadrics with global deformations," *PAMI* **12** (1990).

[9] Pasko, A., Fryazinov, O., Vilbrandt, T., Fayolle, P.-A., and Adzhiev, V., "Procedural function-based modelling of volumetric microstructures," *Graphical Models* **73** (2011).

[10] Talton, J. O., Lou, Y., Lesser, S., Duke, J., Měch, R., and Koltun, V., "Metropolis procedural modeling," *ACM Trans. Graph.* **30** (2011).

[11] Fayolle, P.-A. and Pasko, A., "An evolutionary approach to the extraction of object construction trees from 3D point clouds," *Computer-Aided Design* **74** (2016).

[12] Du, T., Inala, J. P., Pu, Y., Spielberg, A., Schulz, A., Rus, D., Solar-Lezama, A., and Matusik, W., "InverseCSG: Automatic conversion of 3D models to CSG trees," *ACM Trans. Graph.* **37** (2018).

[13] Besl, P. J. and McKay, N. D., "A method for registration of 3-D shapes," *PAMI* **14** (1992).

[14] Fitzgibbon, A. W., "Robust registration of 2D and 3D point sets," *Image Vis. Comput.* **21** (2003).

[15] Mitra, N. J., Gelfand, N., Pottmann, H., and Guibas, L., "Registration of point cloud data from a geometric optimization perspective," in [*Symposium on Geometry Processing*], ACM (2004).

[16] Haugo, S. and Stahl, A., "Iterative closest point with minimal free space constraints," in [*International Symposium on Visual Computing*], Springer (2020).

[17] Haugo, S. and Stahl, A., "Minimal free space constraints for implicit distance bounds," in [*International Symposium on Visual Computing*], Springer (2020).

[18] Tam, G. K., Cheng, Z.-Q., Lai, Y.-K., Langbein, F. C., Liu, Y., Marshall, D., Martin, R. R., Sun, X.-F., and Rosin, P. L., "Registration of 3D point clouds and meshes: a survey from rigid to nonrigid," *TVCG* **19** (2012).

[19] Yezzi, A. and Soatto, S., "Stereoscopic segmentation," *IJCV* **53** (2003).

[20] Xiao, J. and Furukawa, Y., "Reconstructing the world's museums," *IJCV* **110** (2014).

[21] Slavcheva, M., Kehl, W., Navab, N., and Ilic, S., "SDF-2-SDF registration for real-time 3D reconstruction from RGB-D data," *IJCV* **126** (2018).

[22] Whitaker, R. T. and Gregor, J., "A maximum-likelihood surface estimator for dense range data," *PAMI* **24** (2002).

[23] Gargallo, P., Prados, E., and Sturm, P., "Minimizing the reprojection error in surface reconstruction from images," in [*ICCV*], IEEE (2007).

[24] Niemeyer, M., Mescheder, L., Oechsle, M., and Geiger, A., "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," in [*CVPR*], IEEE (2020).

[25] Loper, M. M. and Black, M. J., "Opendr: An approximate differentiable renderer," in [*ECCV*], Springer (2014).

[26] Prisacariu, V. A., Segal, A. V., and Reid, I., "Simultaneous monocular 2D segmentation, 3D pose recovery and 3D reconstruction," in [*ACCV*], Springer (2012).

[27] Liu, S., Zhang, Y., Peng, S., Shi, B., Pollefeys, M., and Cui, Z., "Dist: Rendering deep implicit signed distance function with differentiable sphere tracing," in [*CVPR*], IEEE (2020).

[28] Tagliasacchi, A., Schröder, M., Tkach, A., Bouaziz, S., Botsch, M., and Pauly, M., "Robust articulated-ICP for real-time hand tracking," in [*Computer Graphics Forum*], **34**, Wiley Online Library (2015).

[29] Ma, J., Bae, S. W., and Choi, S., "3D medial axis point approximation using nearest neighbors and the normal field," *The Visual Computer* **28** (2012).