A IMPLEMENTATION DETAILS

We start by describing the speed-ups for proximity queries against the input PLC $\mathcal{T}$ and the set of balls $\mathcal{B}$. Then, we describe the generation of interior samples.

A.1 Supersampling the boundary

The algorithm constructs one k-d tree for each type of strata to speed up proximity queries against $\mathcal{T}$. The k-d tree indexing the sharp corners is simply populated using the set of sharp corners. In order to populate the k-d tree indexing the creases, the algorithm generates a set of 10^5 points sampled uniformly at random from all sharp edges. Similarly, the k-d tree indexing the surface patches is populated using a set of 10^6 points sampled uniformly from all facets. Each generated sample q stores a vector $v_{\sigma,q}$ for each edge or facet $\sigma \ni q$.

A.2 Querying the Boundary k-d trees

Given a point p on a face σ, the algorithm estimates the distance to the nearest non-co-smooth point on the input mesh $\mathcal{T}$ by querying the three boundary k-d trees indexing the sharp corners, creases and surface patches. Let $\mathcal{K}$ denote any of the boundary k-d trees. As the query aims to determine the nearest non-co-smooth point, the co-smoothness test described in Section 2.3 can be used to filter the set of points indexed by $\mathcal{K}$. We implemented a custom k-d tree that performs this filtration on-the-fly. As in the standard k-d tree, the query maintains an estimate of the distance to the nearest point which can be initialized to any sufficiently large value, e.g., the diameter of $\mathcal{T}$ or ∞. By comparing the current estimate against the distance from p to the splitting plane associated with the current node, the query discards an entire subtree if it cannot improve the estimate. The only difference is that due to the filtration defined by the co-smoothness test, a node associated with a point which is co-smooth with p does not provide a distance to update the estimate.

A.3 Ball Neighborhood

To find the set of balls overlapping a given ball b_p, a naive search would be costly. Instead, we find an upper bound on the distance between p and any sample q such that b_q may overlap b_p. Then, we use this bound to query the k-d trees.

Consider two overlapping balls b_p and b_q generated by the MPS procedure, with radii r_p and r_q. W.l.o.g., assume $r_q \geq r_p > 0$. The L-Lipschitzness condition implies that $r_q \leq r_p + L \cdot \|p - q\|$. Since the two ball overlap: $\|p - q\| < r_p + r_q$. Combining the two inequalities, it follows that: $\|p - q\| < r_p + r_q + L \cdot \|p - q\|$. We conclude that $\|p - q\| \leq \frac{2}{1-L} \cdot r_p$. Hence, we query the k-d trees for all balls whose centers are within that distance from p and check if they overlap b_p.

A.4 Point Neighborhood

The deep coverage condition is checked for each new sample p. To speed up this check, we derive an upper bound on the distance between p and the center of any ball that may cover it, and use this to query the k-d trees.

Let q denote the center of the closest ball to p, which we find by a standard nearest-neighbor query to the k-d tree in question. The radius of a ball placed at p respecting L-Lipschitzness can be estimated as $r_p \leq r_q + L \cdot \|p - q\|$.

Consider a ball b_s that barely covers p. It follows that $r_s \leq r_p + L \cdot \|p - s\|$, where $\|p - s\| \leq r_s$. Combining the two inequalities, it follows that $r_s \leq r_q + L \cdot \|p - q\| + L \cdot r_s$, implying $r_s \leq \frac{r_q + L \cdot \|p - q\|}{1 - L}$. Hence, we query the k-d tree for all balls whose centers are within that distance from p and check if they contain p.

A.5 Sampling the interior

The algorithm starts by computing a bounding box bb enclosing the input mesh $\mathcal{T}$; we expand bb to the box 3× larger with the same center. This box is used to initialize the set of interior seeds $S^{\downarrow\downarrow}$ using a lightweight dart-throwing phase. Additional samples are added as needed using the more efficient spoke-darts algorithm [Mitchell et al. 2018]. To guide interior sampling, and ensure a sufficient distance between interior seeds and surface seeds, each surface seed $s \in S^{\updownarrow}$ is assigned a radius r_s by averaging the radii of the three balls in $\mathcal{B}$ defining it. As was done for the set of surface balls $\mathcal{B}$, we maintain two k-d trees $\mathcal{K}^{\updownarrow}$ and $\mathcal{K}^{\downarrow\downarrow}$ for all balls centered at seeds in $S^{\updownarrow}$ or $S^{\downarrow\downarrow}$, respectively.

To initialize $S^{\downarrow\downarrow}$, a new sample z is generated uniformly at random from bb. Then, the closest seed $s \in S^{\updownarrow}$ to z is found by a nearest-neighbor query to $\mathcal{K}^{\updownarrow}$. If $\|z - s\| < r_s$, z is rejected. Otherwise, z gets the label of s and a radius $r_z = r_s + L \cdot \|z - s\|$, which extends the estimated sizing function to the interior of the domain [Miller et al. 1999]. Similarly, the closest interior seed $z^* \in S^{\downarrow\downarrow}$ to z is found by querying $\mathcal{K}^{\downarrow\downarrow}$ and z is rejected if $\|z - z^*\| < r_{z^*}$. Whenever a new sample is rejected, we increment a *miss counter* and otherwise reset it back to 0 if the sample was successfully added into $S^{\downarrow\downarrow}$. Initialization terminates when the miss counter reaches 100.

Then, we continue to add seeds into $S^{\downarrow\downarrow}$ using the spoke-darts algorithm [Mitchell et al. 2018] as follows. We populate a queue Q with all seeds generated by dart-throwing. While the queue is not empty, we pop the next sample z and do the following. Letting b_z be the ball centered at z with radius r_z, we choose a random direction δ and shoot a *spoke* (ray) starting at z in that direction to obtain a new point z_δ at distance $2 \cdot r_z$ from z. Then, we query the k-d trees to find all balls potentially containing z_δ. For each such ball, we trim the line segment ℓ_δ between z and z_δ by pushing z_δ to lie on the boundary of that ball. Once we are done, if z_δ was pushed all the way into the ball b_z, we increment the miss counter. Otherwise, we sample a point z^+ uniformly at random on ℓ_δ, add it as a seed, and reset the miss counter to 0. As before, z^+ is assigned a label and a radius before pushing it into Q. When the miss counter reaches 100, we discard the current point and pop a new point from Q. This process terminates when Q is empty. Finally, we enforce L-Lipschitzness on all interior samples, shrinking balls as necessary, before repopulating Q with all seeds and repeating until no ball gets shrunk.