Title:
Interior Ball-Pivoting on Point Clouds for Offsetting Triangular Meshes

Authors:
Tathagata Chakraborty, tathagata.chakr@hcl.com, HCL Technologies

Keywords:
Offset, Triangular Mesh, Point Cloud, Surface Reconstruction

DOI: 10.14733/cadconfP.2021.83-87

Introduction:

Triangular mesh offsets are useful for hollowing 3D models [7], generating tool paths [5], for clearance analysis and robot path planning among other applications. Smaller offsets are also used to account for shrinkage in casting and rapid prototyping. It is however difficult to compute mesh offsets that are simultaneously defect free and preserve sharp corners.

There are three broad category of techniques for offsetting triangular meshes. In the first category are techniques that obtain the mesh offset by computing the Minkowski sum of the mesh with a sphere of offset radius [14]. A exclusive Minkowski sum approach however requires computing a large number of computationally expensive and numerically unstable booleans. The second category comprises techniques where either the triangles or vertices of the original mesh are translated along a normal to create the offset. In approaches where the triangular faces are moved, they invariably create self-intersections in concave regions and gaps in the convex regions [6], and in techniques where the vertices are moved while maintaining the original topology of the model, the offset becomes increasingly inaccurate as the offset distance increases [12].

A third group of techniques used are those based on volumetric or surface sampling followed by surface reconstruction [11] [14] [8] [9]. In these techniques the offset surface is approximated using a point cloud or a distance field and an appropriate surface reconstruction algorithm is used to triangulate the offset surface. An implicit surface reconstruction technique guarantees a watertight model at the cost of some inaccuracy. Explicit techniques are not very common since they cannot handle noisy point cloud data and do not guarantee a watertight model. In this paper we present a sampling-based method that uses a rarely explored explicit surface reconstruction technique. Our method is fast, preserves sharp corners, and is easy to implement. The major steps of our approach are described briefly in the sections below.

A New Sampling-based Approach:

Our approach consists of first approximating the mesh offset using a uniformly distributed point cloud that has been trimmed to eliminate self-intersections. The trimmed point cloud is then triangulated using the Ball- Pivoting Algorithm (BPA) [1] modified to pivot the ball in the interior of the point cloud and extended to preserve sharp concave corners (see Fig. 1). Our experiments show that the BPA is a very robust algorithm if applied to a uniform point cloud. And while our method does not always produce completely defect-free meshes, it is only because we need a more robust method for estimating the normals at the trim boundary of very sharp corners.

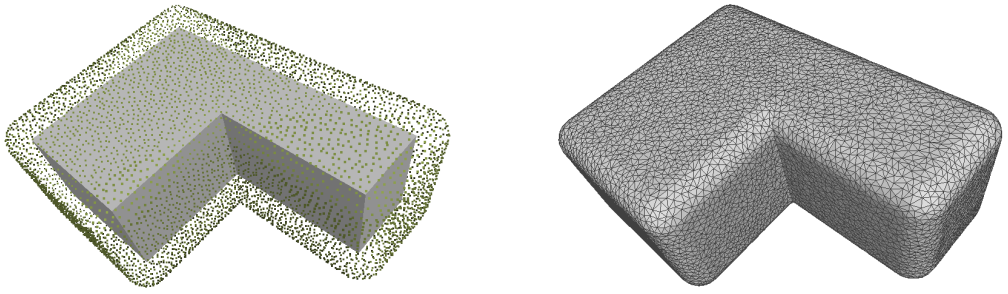

Fig. 1: Left: Original mesh with trimmed point cloud at offset distance; Right: Triangulation of the offset point cloud.

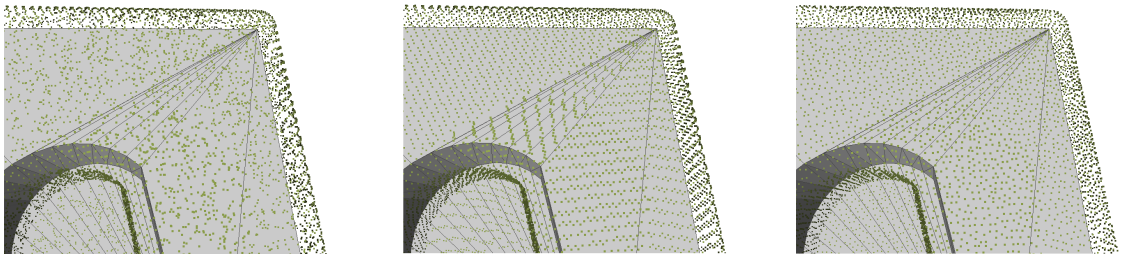

Fig. 2: Left: Point cloud using random point distribution; Center: Point cloud using pseudo-random point distribution; Right: Point cloud using the Poisson-disk decimator.

Uniform Point Cloud Generation:

Our early experiments showed that the BPA works well and produces good and aesthetically pleasing triangulation when the input is a fairly uniform point cloud. It is possible to iteratively apply the BPA using balls of increasing radii to triangulate a non-uniformly distributed point cloud. However in practice it is difficult to determine the range and number of different ball radii to use for a given non-uniform point cloud.

Generating a globally uniform point distribution however is harder than it initially appears. Real-world 3D models are typically non-uniformly tessellated which makes it difficult to ensure a globally uniform point cloud density when sampling each triangle independently (see Fig. 2). Locally this is because the boundary between neighboring triangles get approximated twice and globally because the model is likely to contain very large triangles as well as small and sliver triangles, with the result that there are regions which have a much higher concentration of boundary edges and therefore also a higher density of points.

Generating a uniform point distribution inside a triangle is equally difficult. A random sprinkling of points inside a triangle is far from uniform and not ideal for a triangulation algorithm. A more appropriate method is to use a quasi-random distribution of points [13]. For a globally uniform point distribution though, we first generate a much denser point cloud, trim it to retain only points that are at offset distance, and then decimate it using Poisson-disk sampling.

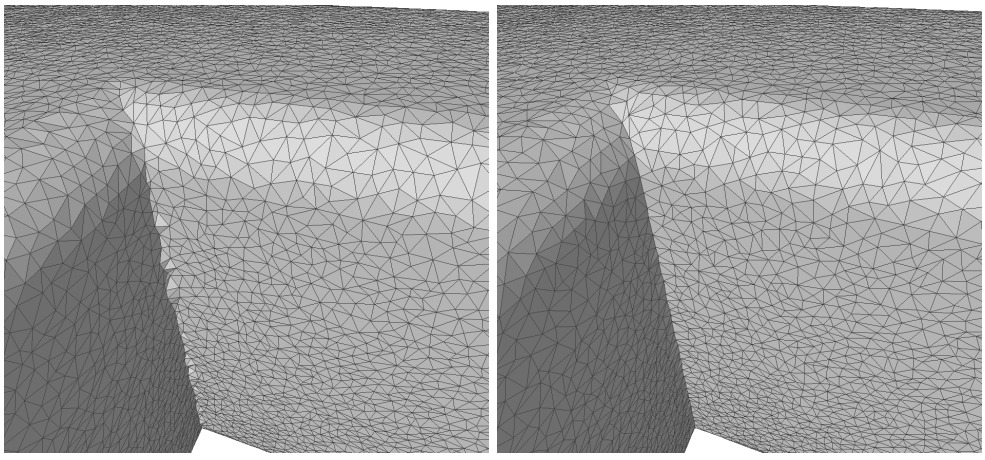

Fig. 3: Left: Triangulation with no trim points at sharp corners; Right: Triangulation with trim points.

Offset Point Cloud Trimming:

After generating the initial offset point cloud, but before decimating it using Poisson-disk sampling, we trim the self-intersecting regions of the point cloud at the concave corners and near the boundaries of the patch surfaces. In order to trim the offset point cloud of self-intersections we check each point and find the minimum distance from the point to the original mesh. To do this we overlay a voxel grid of offset distance resolution on the original triangulated mesh, where each voxel stores the list of triangles which pass through or intersect that voxel. For a given point in the offset point cloud, we first find the voxel in which it located, and then find all the triangles in the 1-voxel neighborhood of this voxel. Next we compute analytically the minimum distance of the offset point under consideration to each of the triangles found in the 1-voxel neighborhood and if the minimum of these distances is less than the offset distance we discard the point and mark it as invalid.

Preserving Sharp Corners:

For each invalid point found during trimming, we explore the neighborhood of the point to see if we can find a nearby valid point that lies on the trim boundary. Finding points on the trim boundary is essential for preserving sharp concave features in the offset. Without the presence of these trim boundary points and a surface reconstruction method that includes these points the triangulation in the sharp corners would be very jagged (see Fig. 3). Jagged corners not only add to the inaccuracy of the offset, they are also problematic in many domains like manufacturing and for tool path generation.

To find the trim boundary points we first compute two orthonormal vectors lying on the offset surface at each invalid point. Next, at the point cloud resolution distance along each of these orthonormal directions we find two points, one in either direction. If any of these four points are valid, we do a binary search between the invalid point and the valid point to find the nearest valid point to the invalid point. This point is then added to the point cloud.

Point Decimation by Poisson-disk Sampling:

As noted earlier, it is very difficult to generate a globally uniform point cloud directly by random sampling on the mesh (see Fig. 2). We therefore initially generate a much denser point cloud using quasi-random sampling, and then decimate the points using Poisson-disk sampling to get a globally uniform distribution.

We implement Poisson-disk sampling using a voxel map constructed at the desired point cloud reso-

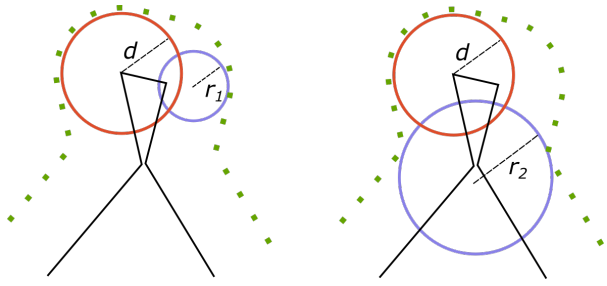

Fig. 4: BPA ball radius r must be less than offset distance d so that it can access all points.

lution r. We then iterate randomly through all the voxels and for each non-empty voxel select a random point from the voxel as a candidate point to preserve. We then find and discard all other points in the r-neighborhood of the candidate point (i.e. 1-voxel neighborhood in the voxel map).

Triangulation using the Modified BPA:

The Ball-Pivoting Algorithm (BPA) [1] [4] is an elegant method for triangulating a set of non-noisy points, and thus ideally suited for our purpose where the point cloud is artificially generated. Our choice of the BPA was also driven by the ease with which we could extend the algorithm by adding our own heuristic for the preservation of sharp concave corners.

Our modifications to the BPA are trivial but nonetheless very important. First, instead of rolling the ball outside the point cloud we roll it inside the point cloud. This choice forces us to use a ball radius less than the offset distance and also a denser point cloud (see Fig. 4). Rolling the ball inside point cloud enables us to properly triangulate narrow channels and gaps which the standard BPA (and most other implicit and explicit surface reconstruction technique) cannot handle.

Initially we explored heuristic relaxation of various checks in the BPA to preserve sharp concave corners. However, further research indicated that all these heuristics could be eliminated if only the normals at the trim boundary point could be accurately estimated. We therefore developed various methods to estimate the normal at the trim points by exploring the points in the neighborhood of the trim point. We note here that accurate normal estimation at the trim points is not always easy especially at very sharp concave corners and in the absence of proper trim normals some relaxation heuristics will be required.

Conclusions:

Offsetting triangular meshes is a difficult problem to solve robustly. Taking a Minkowski sum approach offers a possible solution, but depends upon the availability of a robust mesh boolean implementation, which is an equally difficult problem to solve. Taking a volumetric approach where the offset is approximated by a signed distance field and then using an implicit surface reconstruction technique is probably the most common technique used not only for surface reconstruction in general but also for estimating offsets. These volumetric approaches however do not scale well because the memory required to store the volumetric data increases exponentially with model size, and a coarser voxellization reduces the accuracy of the reconstruction. In this paper we have described an unorthodox approach for computing mesh offsets using the Ball-Pivoting Algorithm which is a relatively uncommon explicit surface reconstruction technique. We have shown how the method can be extended to preserve narrow gaps and sharp concave corners in the offset. Our results indicate that the method performs robustly as long as the normals at the sharp corners are robustly estimated. Our method does not require much memory, however performance remains a problem for large models and a robust normal estimation technique needs to be found.

Acknowledgement:
This research was undertaken as a part of the CAMWorks project. CAMWorks is a popular CAM software used by small and medium sized manufacturing workshops and industries. Thanks to Vivek Govekar and Nitin Umap, product manager and project manager respectively for CAMWorks for providing us the opportunity to spend the last year exploring and prototyping the research presented in this paper. To Swadhin Bhide thanks for suggesting various research directions, and thanks to Hariharan Krishnamurthy for reviewing several drafts of the paper.

References:

[1] Bernardini, F.; Mittleman, J.; Rushmeier, H.; Silva, C.; Taubin, G.: The ball-pivoting algorithm for surface reconstruction, IEEE transactions on visualization and computer graphics, 5(4), 349-359, 1999. https://doi.org/10.1109/2945.817351

[2] CGAL 5.2, 3D Minkowski Sum of Polyhedra. https://doc.cgal.org/latest/Minkowski_sum_3/index.html

[3] Chen, Y.; Wang, H.; Rosen, D. W.; Rossignac, J.: A point-based offsetting method of polygonal meshes, ASME Journal of Computing and Information Science in Engineering, 1-21, 2005.

[4] Digne, J.: An analysis and implementation of a parallel ball pivoting algorithm, Image Processing On Line, 4, 149-168, 2014. https://doi.org/10.5201/ipol.2014.81

[5] Kim, S. J.; Yang, M. Y.: Triangular mesh offset for generalized cutter, Computer-Aided Design, 37(10), 999-1014, 2005. https://doi.org/10.1016/j.cad.2004.10.002

[6] Kim, S. J.; Lee, D. Y.; Yang, M. Y.: Offset triangular mesh using the multiple normal vectors of a vertex, Computer-Aided Design and Applications, 1(1-4), 285-291, 2004. https://doi.org/10.1080/16864360.2004.10738269

[7] Koc, B.; Lee, Y. S.: Hollowing STL Objects with Biarcs Fitting to Improve Efficiency and Accuracy of Rapid Prototyping Processes, In IIE Annual Conference. Proceedings (p. 1), Institute of Industrial and Systems Engineers (IISE), 2002.

[8] Lien, J. M.: Covering Minkowski sum boundary using points with applications. Computer Aided Geometric Design, 25(8), 652-666. 2008. https://doi.org/10.1016/j.cagd.2008.06.006

[9] Liu, S.; Wang, C. C.: Fast intersection-free offset surface generation from freeform models with triangular meshes. IEEE Transactions on Automation Science and Engineering, 8(2), 347-360. 2010. https://doi.org/10.1109/TASE.2010.2066563

[10] Malosio, M.; Pedrocchi, N.; Tosatti, L. M.: Algorithm to offset and smooth tessellated surfaces, Computer-aided design and applications, 6(3), 351-363, 2009. https://doi.org/10.3722/cadaps.2009.351-363

[11] Pavić, D.; Kobbelt, L.: High-resolution volumetric computation of offset surfaces with feature preservation, In Computer Graphics Forum (Vol. 27, No. 2, pp. 165-174), Oxford, UK: Blackwell Publishing Ltd., 2008. https://doi.org/10.1111/j.1467-8659.2008.01113.x

[12] Qu, X; Stucker, B.: A 3D surface offset method for STL-format models. Rapid prototyping journal. 2003. https://doi.org/10.1108/13552540310477436

[13] Roberts, M.: Evenly Distributing Points in a Triangle. http://extremelearning.com.au/evenly-distributing-points-in-a-triangle

[14] Varadhan, G.; Manocha, D.: Accurate Minkowski sum approximation of polyhedral models. In 12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings. (pp. 392-401). IEEE.