# Dynamic Subdivision Sculpting

Raul Fernandez Hernandez*
Hermanos Saz University

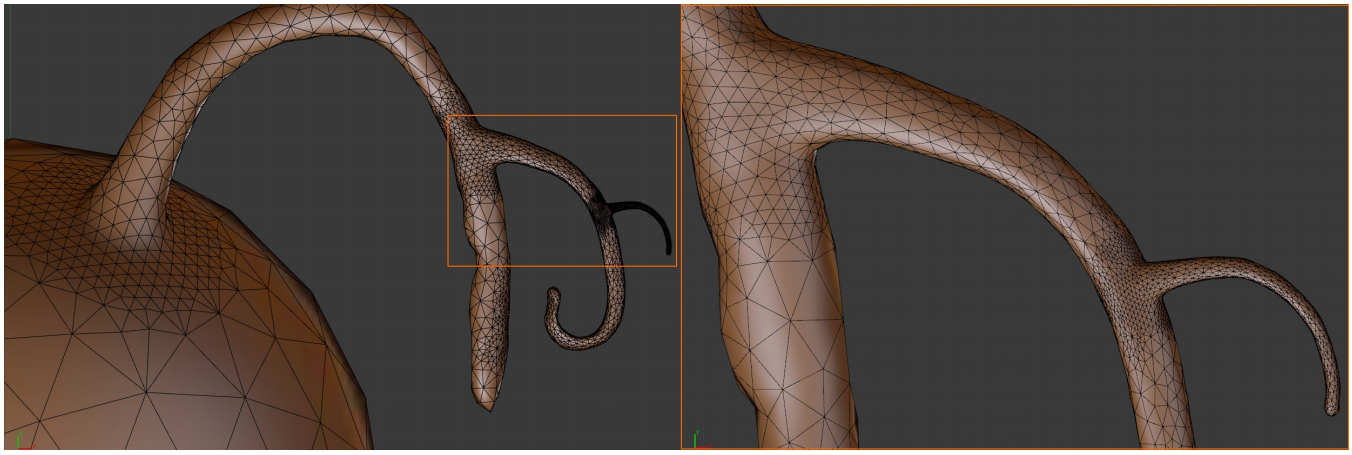

**Figure 1:** *Dynamic subdivision sculpting.*

## Abstract

Sculpting is an important field in computer graphics and one of the most advanced and used techniques to create 3D models due to its intuitive and easily understood way of interacting with 3D objects, traditional sculpting pipelines suffer from several drawbacks related to the progressive loss of detail as the model is sculpted. Generally the surface area becomes expanded and polygons stretched, moreover, in order to sculpt fine detail zones the model must be subdivided to the desired detail level. This comes at the cost of excessively subdividing areas that only require relatively low levels of detail (LOD) decreasing performance and increasing resource cost.

In this paper we introduce the theory of dynamic operator sculpting. This is applied to a dynamic detail subdivision sculpting algorithm that can be integrated into any sculpting pipeline. Sculpting subdivision refinement is an appealing way to add detail only where it is needed and with the optimal mesh density, it also solves all of the previously mentioned issues with traditional sculpting approach and ensures a user defined level of detail under the brush stroke, regardless of the distance of the mesh from the camera in the viewport.

**CR Categories:** K.6.1 [Computer Graphics]: Polygonal sculpting—; K.6.2 [Computer Graphics]: Mesh generation

**Keywords:** dynamic subdivision, sculpting, operator.

## 1 Introduction

There's several ways to create an object in 3D, each of which have strengths and weaknesses. They can be classified into two main groups: direct and raw methods. Direct methods are topology aware and construct the objects with direct control over mesh elements (verts, edges, faces). This method has the advantage of giving full control over the final mesh topology at the expense of much slower modeling times. Raw methods, on the other hand, operate higher in the stack, with no direct control over mesh components. Examples include 3D scanning reconstruction and digital sculpting. Raw methods offer much more rapid creation of the overall desired shape albeit with far less control over topology. In practice, both methods are often combined to try and get some of the speed advantages of raw methods while still providing the specific topological features required by most animation pipelines.

The evolution of digital modeling programs has led to life-like sculpting workflows, giving the artist an intuitive way to shape the model similar to working with familiar materials such as clay.

However, traditional digital sculpting algorithms suffer from several issues. Firstly, if the geometry detail under the brush is of relatively low frequency, not high enough to sample the brush stroke function then the mesh cannot adequately sample the brush. Secondly, as the artist sculpts repeatedly over the same area, the underlying polygons tend to stretch and eventually become larger, decreasing detail frequency (fig 2) reinforcing the first problem. Third, when small detailed features need to be added to the model, the entire object needs to be refined to match the desired level of detail, leading to huge increase in wasted polygons in areas where very coarse detail is actually needed.

In this paper we propose an adaptive, dynamic subdivision sculpting algorithm that can be integrated into any sculpting pipeline. Many recent applications are developing such features in their sculpting tool-set but so far no overall theory has been proposed. We define here a theoretical framework that will allow the definition of a dynamic subdivision sculpting algorithm and set the foundations for future development in this field. In section 2 we summarize previous work in the dynamic refinement sculpting and painting fields. In section 3 we define sculpting and subdivision over the

*e-mail: farsthary84@gmaill.com

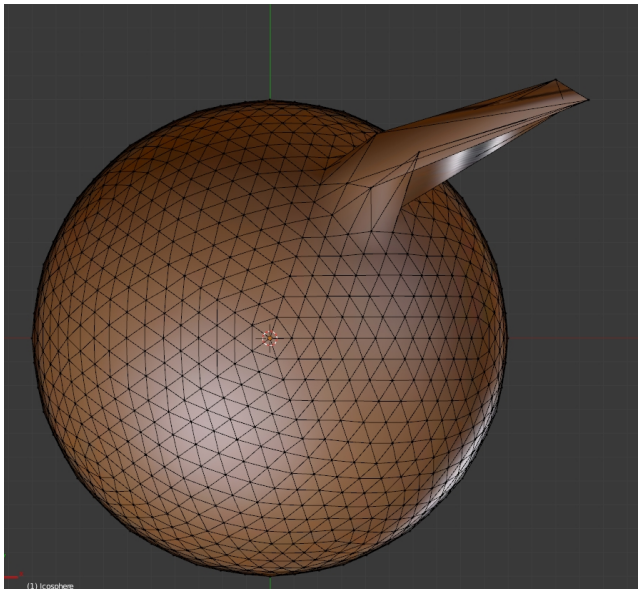

**Figure 2:** *Low detail model traditionally sculpted.*

base mesh as operators. Our algorithm is exposed in subsequent sections and we conclude in section 4 examining the direction of future potential development.

## 2 Related Work

An extensive dynamic subdivision sculpting is beyond the scope of this paper however dynamic subdivision sculpting on polygonal meshes is a relatively new field in computer graphics and is mainly developed as an internal technique with very little shared knowledge among implementations.

[MCDONNELL and HONGQIN 2002] develop a sculpting system focused on the physical properties of a virtual clay and achieving dynamic subdivision geometry levels using Catmull-Clark subdivision scheme [E.CATMULL and J.CLARK 1978]. The dynamic subdivision here is achieved through the extrude tool defined in his research however the scheme does not have a local refinement approach. It is similar to multiresolution sculpting , which can dynamically change subdivision levels but not in a local way. It is not adaptive geometric detail sculpting, just an adaptive level of detail where changes in lower levels are propagated to areas of higher detail/refinement.

There has been other development in dynamic sculpting [WANG and KAUFMAN 1995], [SHEKHAR et al. 1996] and [HO et al. 2004] but seemingly not on polygonal meshes. For example, volumetric sculpting outputs polygonal meshes however internally it employs other data types such as volumetric cells organized in arrays, octrees, etc.

Some refinement schemes have also been employed in [BOIER-MARTIN et al. 2005] related to boolean operators. The authors employ a subdivision scheme in order to approximate and fit the resultant boolean mesh to the theoretical surface result.

Multiresolution sculpting is widely used in Blender [ROSENDAAL 1995] by software and in [RITSCHEL et al. 2006] a GPU accelerated algorithm is employed. Again, they aim at static sculpting methods as the algorithm knows all the detail levels the user is sculpting at run time. In [ZORIN et al. 2000] an interesting multiresolution algorithm is presented with underlying algorithms for refinement and detail reduction, which enables making them local and adaptive and thereby considerably improving their efficiency. This could be considered a hybrid which employs many of the features of the dynamic subdivision sculpting algorithm. A physically based painting and sculpting algorithm with dynamic subdivision is presented in [LAWRENCE and FUNKHOUSER 2003]. This includes mesh support as well as volumetric models. The physical nature of the sculpting interface though, does not integrate easily with a generic sculpting pipeline and the update of the model is performed in a simulation step after the stroke is executed, thus lacking any real time subdivision update or instant user feedback.

Voxel sculpting or volumetric sculpting methods naturally implement dynamic refinement algorithm as in [MCDONNELL et al. 2001]. Voxels are recursively subdivided to reach a specific LOD, and the resultant mesh is a reconstruction from the volume data. There has been a lot of research and applications developed in this field.

Starting in 2009 [PETTERSON 2009] developed Sculptris, which was based on a dynamic subdivision tool. Due to its intuitive design and fast results it quickly became a reference in the field, and forms the technical inspiration of this paper.

Our research has lead to the implementation of the presented algorithms in this paper into Blender pipeline.

## 3 Dynamic Operator Sculpting

Lets define a Mesh Operator as a transformation applied to an input mesh that then outputs another mesh. A Mesh is a group of polygonal geometric elements such as vertices, edges and faces. Each of these can include any amount of properties or parameters depending on each individual use case, such as color, weight, selection, etc, with further topological relationships amongst them.

Sculpting (fig 3) is a local set of operators, having a finite support defined by a brush radius which applies topological transformation to mesh elements inside the brush radius. Sculpting itself does not change topology. It only transforms pre-existing elements.

Subdivision (fig 4) is also a local operator with a finite support defined as the selected mesh elements domain. Subdivision increases the frequency component changing topology by adding new mesh elements. There are several types of subdivision schemes but lets focus on Loop [LOOP 1987] triangle subdivision scheme, for its simplicity and low computational cost. This is also worth examining because of the fact that any robust subdivision scheme could be used because the output mesh will always converge to the correct solution.

Decimation (fig 5) is defined as the inverse of the subdivision operator. It is also a local operator with a finite support range defined as the selected domain of mesh elements and is used to obtain a low frequency component, so the output mesh has fewer elements than the original. A decimation operator can be implemented in many ways.

Decimation is not a convergent operator, so practical results may depend on the actual mesh simplification algorithm implemented however it does not change the essence of the algorithm developed in this paper.

Relaxation (fig 6) is defined as an operator with a finite support defined by the selected domain of mesh elements that transforms existing elements according to a defined smoothing /relaxation criteria. Many relaxation algorithms have being proposed and since

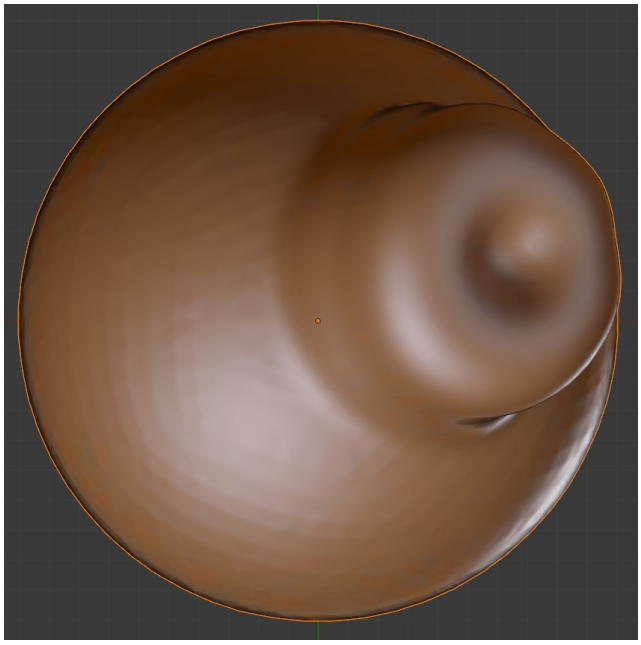

**Figure 3:** *Sculpting operator.*

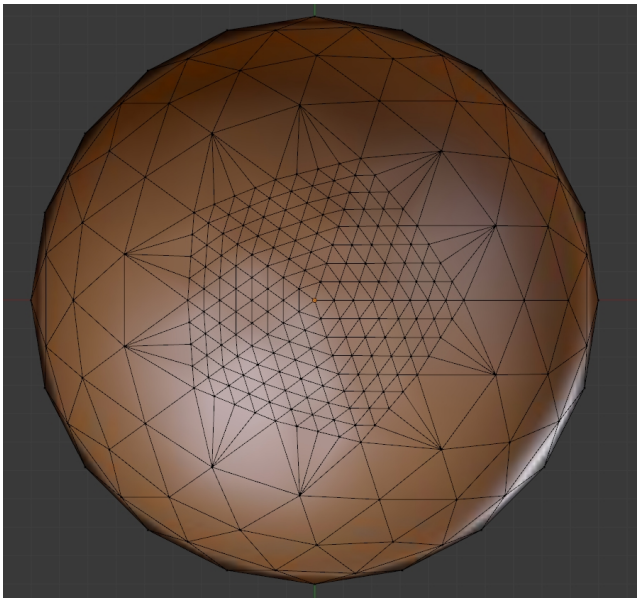

**Figure 4:** *Subdivision operator.*

they are generally convergent and stable most could be used interchangeably in the proposed algorithm.

In general, any mesh operator with a finite support could be used in combination with the sculpt operator. The result is a convolution operator with a finite support as well. In practice operators could have infinite support, but then the advantage of sculpting localized operator features is lost. The result operator could again be convoluted with other mesh operators in any combination to achieve more advanced operator tools.

The essence of our algorithm is the Sculpt operator convolution equation:

$$m = (S \otimes P)(m) \tag{1}$$

Where S is the sculpt operator over a mesh m, P is any other mesh operator, particularly the subdivision operator, and R is the resultant convoluted mesh operator.

The key factor is to link the sculpt domain and the operator domain with some function or algorithm. The actual implementation of any convolution sculpting operator algorithm depends on the convoluted operator. For that reason in this paper we will focus on the subdivision operator due to its importance and also because it represents a general family of topology changing operators, however any of the others could be used in place of subdivision operator.

### 3.1 Dynamic Subdivision Sculpting

A generic static sculpting pipeline operates as follows:

**On each bush stroke step:**

**1. Select mesh elements inside brush sphere (set sculpt domain)**
**2. Apply sculpt operator (transform elements according to brush settings)**

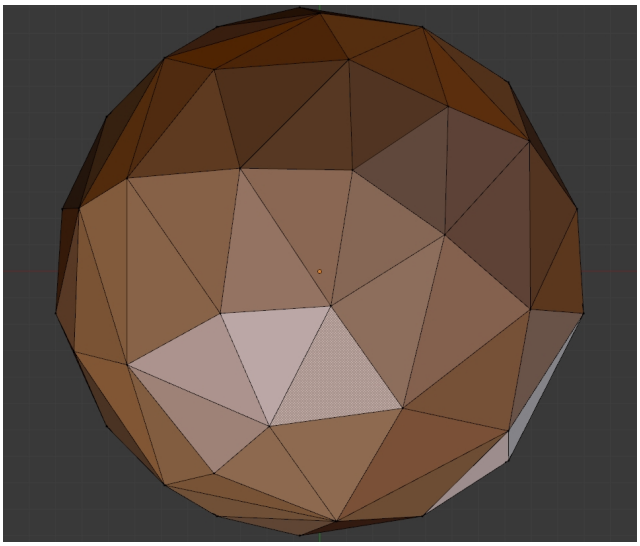

**Figure 5:** *Decimation operator.*

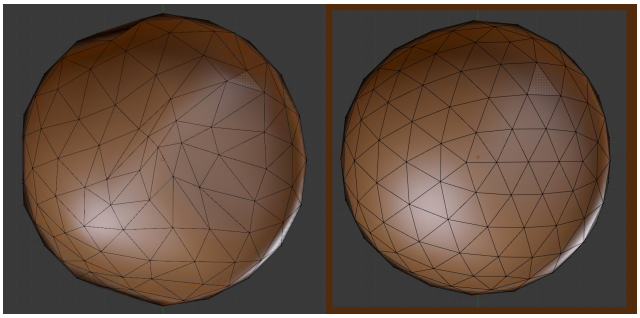

**Figure 6:** *Relaxation operator.*

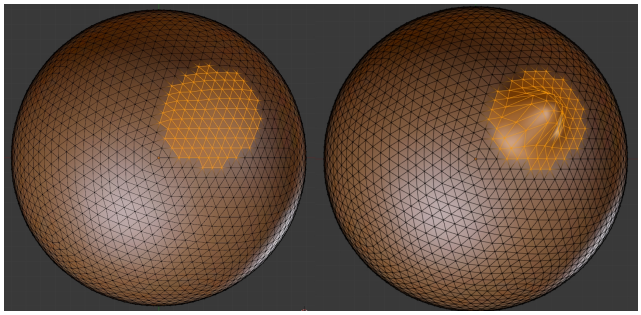

**Figure 7:** *Static sculpting steps.*

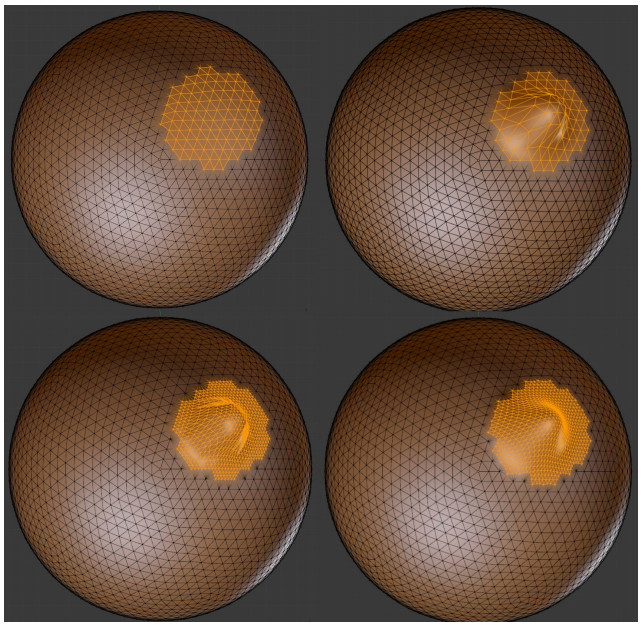

**Figure 8:** *Dynamic subdivision sculpting basic steps: domain selection, sculpting, subdivision, relaxation.*

Step 1 can be performed in many ways, using brute force methods such as raycast, kd-tree, etc. to test if mesh elements intersect with the brush sphere. Using a spatial partition data structure however, allows far more efficient sculpting of high polygon meshes. It does do by reducing the amount of intersection tests that are a critical part in the performance of a sculpting pipeline.

Our algorithm extends any generic sculpting pipeline as follows:

**On each bush stroke step:**

**1. Select mesh elements inside brush sphere (set sculpt domain)**
**2. Select subdivision operator elements from sculpt domain (set operator domain)**
**3. Select relaxation operator elements from subdivision domain (set operator domain)**
**4. Apply sculpt operator (transform elements according to brush settings)**
**5. Apply subdivision operator (subdivide elements according to operator settings)**
**6. Apply relaxation operator (smooth elements according to operator settings)**

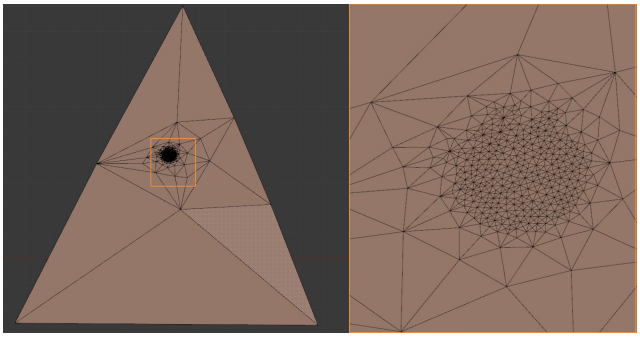

**Figure 9:** *Big face subdivision.*

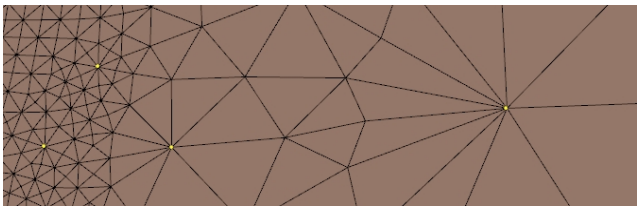

**Figure 10:** *Pole vertices.*

Step 1 performs an intersection test of each mesh element against the brush sphere radius. This is used to apply the sculpt operator. The selection criterion is whether or not any part of the mesh elements fall inside brush sphere radius. It is very important to include every face at least partially inside the brush sphere, so that if the size of the face is much larger than the size of the brush radius, it will still correctly be subdivided and sculpted.

Step 2 needs special attention, as the sculpt domain need not necessarily be the same as the subdivision domain. In fact, it shouldnt be the same in order to obtain clean topology. As the brush moves over the mesh, new subdivided vertices can quickly become poles or extraordinary vertices (fig 10) with very high valence, destroying the topology smoothness. For that reason a special brush boundary treatment is made, taking more elements selection derivatives out of the sculpt domain.

Our algorithm includes in the subdivision operator domain pole vertices of valence higher than a defined threshold in a vertex ring up to second order out of sculpt domain. For triangle meshes, any value higher than 6 could be considered a pole. A threshold of 9 edges per vertex is selected in our algorithm. This way poles and extraordinary vertices are quickly detected and isolated preventing them from growing much more and improving topology quality. So the subdivision domain consists of the union of the sculpt domain with all of the extended boundary faces that have extraordinary vertices on them. (fig 11)

The selection criterion for the convoluted operator is very specific to the operator. For the Subdivide operator several split criteria could be used and the output mesh quality will depend heavily on which one is selected.

The simplest split condition is to test for edges is longer than a LOD threshold. When found, the triangle face is divided into two new triangle faces. Since that split criteria no longer satisfies the Loop subdivision scheme, which is a uniform scheme a special set of split rules needs to be defined for extraordinary and pole vertices. In this case, the general the quality of the mesh will not be as uniform either.

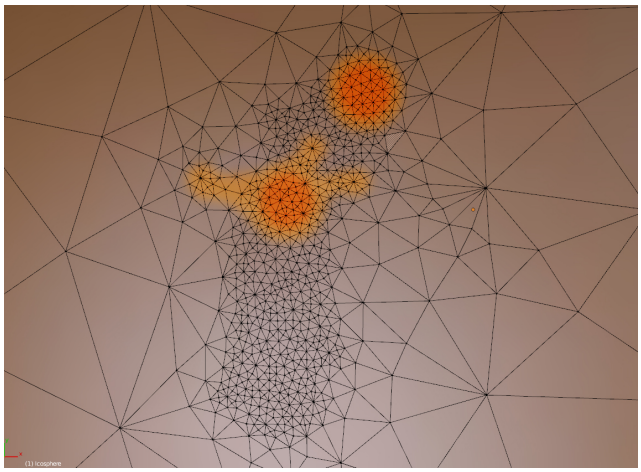

**Figure 11:** *Sculpting operator domain (dark orange) and extended subdivision operator domain (light orange).*

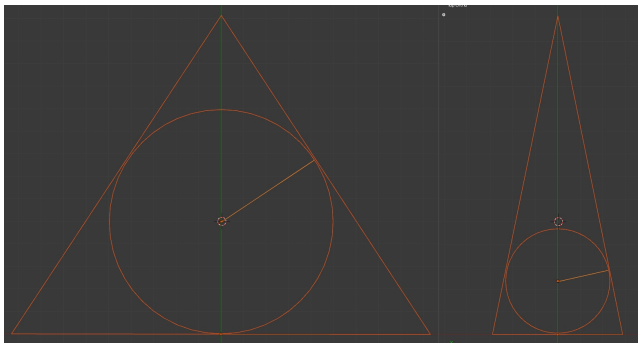

**Figure 12:** *Triangle faces inscribed circle and circle radius.*

A uniform split criteria that is compatible with the Loop scheme is preferable, which means splitting the whole edges of the face. Another criterion could be to compare the perimeter of the triangle or the area of the face, both of which could lead to thin, stretched triangles.

Our research found that using the in-circle triangle face radius (fig 12) as the split criteria provided the best results as long, thin, stretched triangles for a given LOD are avoided (fig 13).

Step 3 can reuse the previous domain so no new calculations need to be performed here.

Step 5 for the subdivision operator we use the Loop scheme (fig 14), splitting in half each mesh edge into two edges, and each triangle face into four triangle faces.

Step 6 for the relaxation operator we use a modified laplacian relaxation algorithm with projected back displacements in the vertex normal direction to avoid shape deformation with successive brush strokes. The laplacian relaxation algorithm displaces the vertices position toward the average neighboring vertices centroid.

## 4 Implementation

We implemented the algorithm integrating into the Blender sculpting pipeline as can be seen in [BART 2011]. The existing sculpting scheme uses raycast as mouse collision detection with the model that traverses a PBVH spatial partition tree. We select different

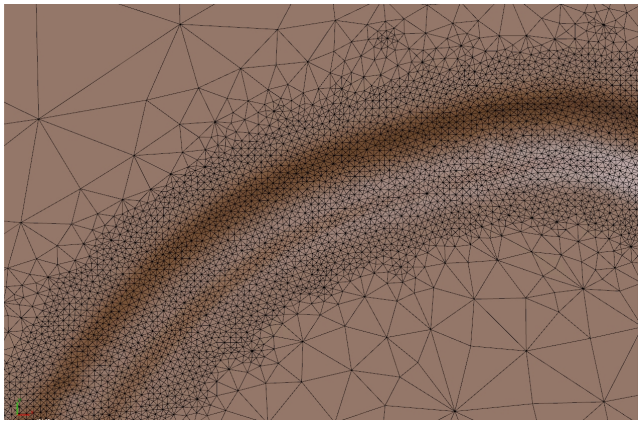

**Figure 13:** *Incircle split criteria result in cleaner meshes.*

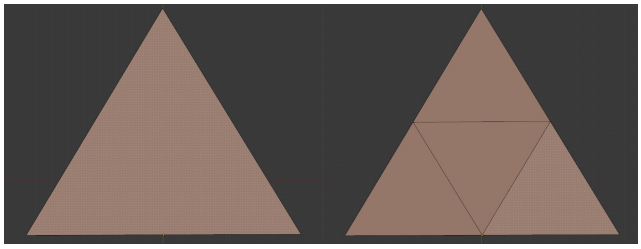

**Figure 14:** *Loop subdivision scheme.*

domain elements using flags and iterating over tree leaf nodes elements. An important aspect of the implementation is that each mesh element is linked to the corresponding hierarchies lower element types. A face has direct pointers to edges and vertices while edges have direct pointers to the vertices only.

The subdivision operator and the smoothing operator introduce a performance penalty over a static sculpting pipeline; however the degree of this depends heavily on the implementation and optimizations of each. This is why dynamic subdivision sculpting cannot completely replace a static pipeline that is taking advantage of preallocated elements. In high polygon count models with a uniform mesh density (fig 15); dynamic subdivision sculpting is not recommended. Its advantage over a traditional approach depends on models with low to medium detail and / or non uniform mesh density (fig 16).

## 5 Conclusion and Future Work

We have introduced the mathematical background of sculpting convolution operators, that previously where tackled in an empirical and technical way, and have shown an easy to implement algorithm to integrate dynamic subdivision and relaxation into any existing sculpting pipeline. The method could also be extended to include any other mesh operators to build more advanced tools still. A very interesting extension would be to add support for mesh detail reduction or decimation. The ability of the user to remove detail in addition to adding it, would allow the user to control and improve the final adaptive mesh result.

This algorithm could also be extended by swapping the sculpt operator with any other user controlled operator with a finite range similar to painting, as can be seen in [FU and CHEN 2008].

Finally, further optimizations could include exchanging the use of

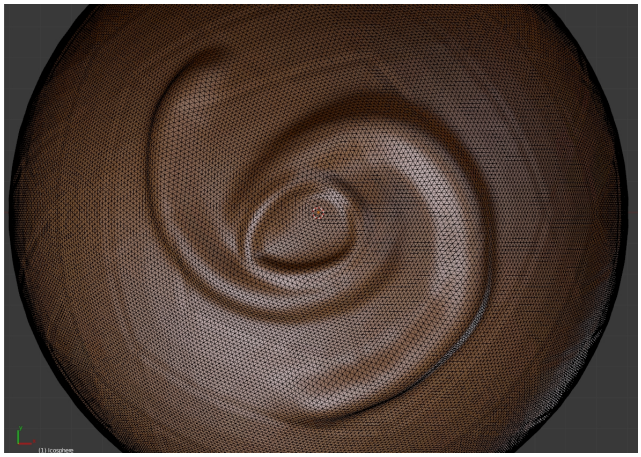

**Figure 15:** *In high detail, uniform model, greater performances are found in traditional static sculpting methods.*

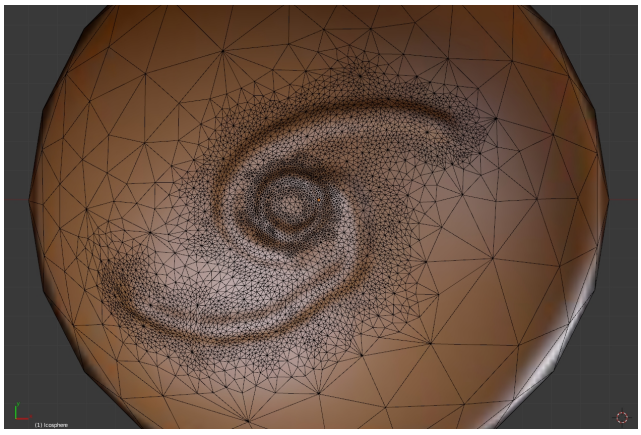

**Figure 16:** *In low detail, non uniform models, dynamic subdivision sculpting algorithms perform better.*

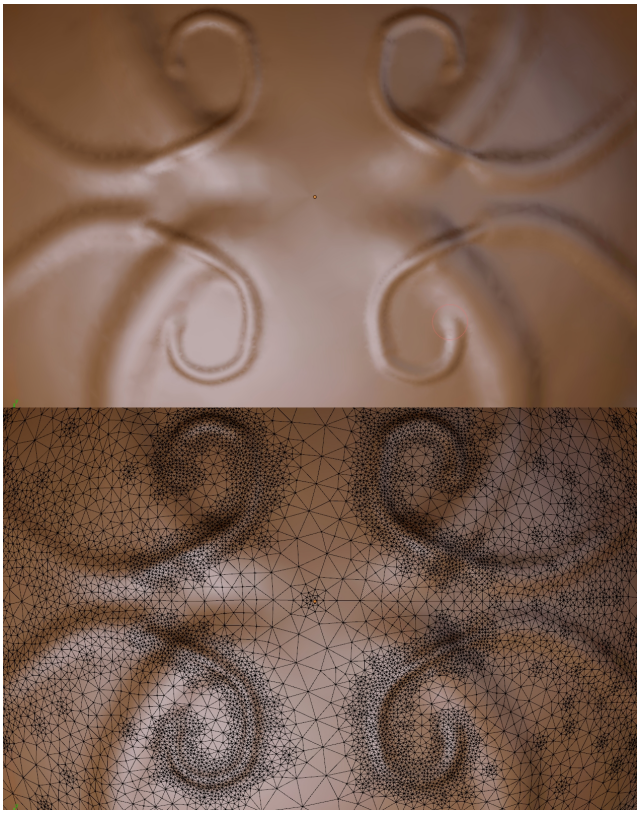

**Figure 17:** *Subdivision strokes on mesh.*

a flag as the domain, with linked lists to process only the needed geometry for each operator. This way brush radius support can become the primary performance concern rather than polygon count.

## Acknowledgements

The Author wishes to thank everyone who was involved in making this paper possible, especially to Piers and Sheryl Duruz for his restless efforts.

## References

BART, 2011. Toward-sculptris-like-sculpting.

BOIERMARTIN, I., ZORIN, D., AND BERNARDINI, F. 2005. A survey of subdivisionbased tools for surface modeling. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 12–14.

E.CATMULL, AND J.CLARK. 1978. Recursively generated bspline surfaces on arbitrary topology. *ComputerAided Design 10*, 6, 350–355.

FU, Y., AND CHEN, Y. 2008. Haptic 3d mesh painting based on dynamic subdivision. *ComputerAided Design and Applications 5*, 131–141.

HO, C., LU, Y., LIN, H., GUAN, S., CHO, S., LIANG, R., CHEN, B., AND OUHYOUNG, M. 2004. Feature refinement strategy for extended marching cubes: Handling on dynamic nature of realtime sculpting application. *IEEE International Conference Multimedia and Expo (ICME)*, 2–3.

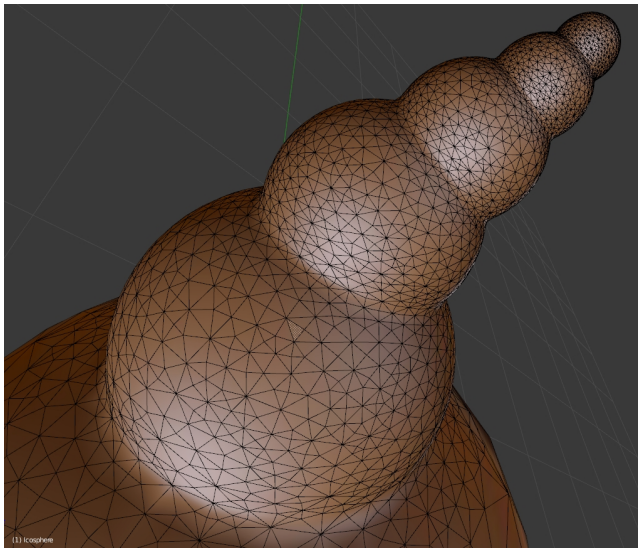

**Figure 18:** *Adaptive detail over sculpted mesh.*


LAWRENCE, J., AND FUNKHOUSER, T. 2003. A painting interface for interactive surface deformations. *11th Pacific Conference on Computer Graphics and Applications*, 141–151.

LOOP, C. 1987. *Smooth subdivision surfaces based on triangles*. PhD thesis, University of Utah.

MCDONNELL, K. T., AND HONGQIN. 2002. Dynamic sculpting and animation of freeform subdivision solids. *Proc. of SIGGRAPH*, 1–8.

MCDONNELL, K. T., QIN, H., AND WLODARCZYK, R. A. 2001. Virtual clay: A realtime sculpting system with haptic toolkits. *Proceedings of the 2001 ACM Symposium on Interactive 3D Graphics*, 179–190.

PETTERSON, T., 2009. Sculptris[software].

RITSCHEL, T., BOTSCH, M., AND MLLER, S. 2006. Multiresolution gpu mesh painting. *Eurographics 2006*, 1–4.

ROSENDAAL, T., 1995. Blender[software].

SHEKHAR, R., FAYYAD, E., YAGEL, R., AND CORNHILL, J. F. 1996. Octreebased decimation of marching cubes surfaces. *Proc.of IEEE Visualization*, 335–342.

WANG, S. W., AND KAUFMAN, A. E. 1995. Volume sculpting. *Proc. of ACM Symposium on Interactive 3D Graphics*, 151–156.

ZORIN, D., SCHRODER, P., AND SWELDENS, W. 2000. Interactive multiresolution mesh editing. *SIGGRAPH 2000*, 1–10.