DOI: 10.1111/cgf.12437
Eurographics Symposium on Geometry Processing 2014
Thomas Funkhouser and Shi-Min Hu
(Guest Editors)

Volume 33 (2014), Number 5

Clever Support: Efficient Support Structure Generation for Digital Fabrication

J. Vanek[1] and J. A. G. Galicia[1] and B. Benes[1]

[1]Purdue University

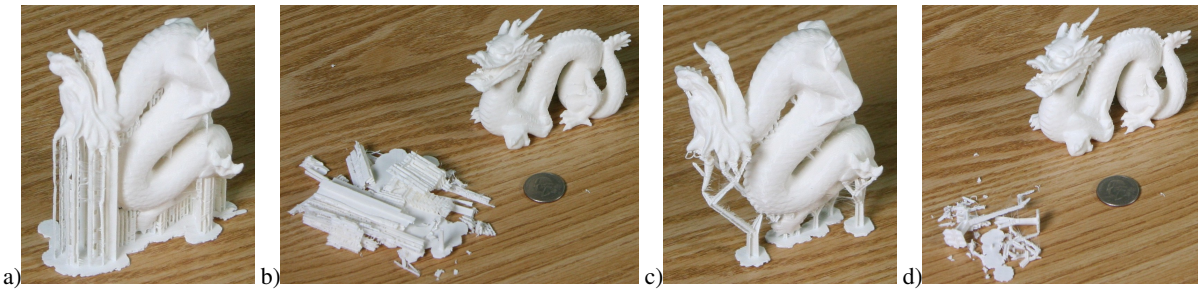

Figure 1: *The support material generated by the built-in 3D printing software for MakerBot® Replicator™ 2 a) and the amount of support material b). Our solution c) reduces the amount of the support material d), leads to faster printing, and higher quality of the fabricated model.*

Abstract

We introduce an optimization framework for the reduction of support structures required by 3D printers based on Fused Deposition Modeling (FDM) technology. The printers need to connect overhangs with the lower parts of the object or the ground in order to print them. Since the support material needs to be printed first and discarded later, optimizing its volume can lead to material and printing time savings. We present a novel, geometry-based approach that minimizes the support material while providing sufficient support. Using our approach, the input 3D model is first oriented into a position with minimal area that requires support. Then the points in this area that require support are detected. For these points the supporting structure is progressively built while attempting to minimize the overall length of the support structure. The resulting structure has a tree-like shape that effectively supports the overhangs. We have tested our algorithm on the MakerBot® Replicator™ 2 printer and we compared our solution to the embedded software solution in this printer and to Autodesk® Meshmixer™ software. Our solution reduced printing time by an average of 29.4% (ranging from 13.9% to 49.5%) and the amount of material by 40.5% (ranging from 24.5% to 68.1%).

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—I.3.8 [Computer Graphics]: Applications—

1. Introduction

Three-dimensional (3D) printers can fabricate virtually any shape captured by a 3D model. They can create models from various materials and they can even combine multiple materials together. Today's 3D printers typically fabricate the object by additive layering of a material that is fused together layer by layer. Because of the lower prices of 3D printers and materials, digital fabrication is becoming available to the general public. In this work we will focus mainly on

Fused Deposition Modeling (FDM) printers since they are very common, inexpensive, and widely used.

The printing process starts from the bottom of the printing tray and continues successively layer by layer to the top. This causes problems for overhangs that cannot be printed because there is no supporting layer beneath them. The 3D printers based on FDM and Stereolitography (SLA) technology solve this problem by creating support structures for the overhanging parts (see Figure 1 a)). The support structures need to be manually discarded after the printing. Some printers use cheaper material for the supporting structures but in other cases the main printing material is used. For example, the FDM printers use the main printing material for support structures. The support material adds cost to the printing. For SLA printers, the price of the support material can be almost half of the price of the primary material.

It is beneficial to minimize the amount of support material. Less supports contribute to the faster printing time, less cleaning of the 3D model, and reduces the overall price of the object. Existing support generators, usually built-in within the 3D printer software, do not produce optimal supporting structure and the amount of extra material required for the supports is unnecessary high as shown in Figure 1 b. The supports are usually inserted as vertical columns and they connect the overhangs with the nearest underlying point on the printing tray or on the model itself.

We present a geometry-based solution that attempts to minimize the amount of supports without compromising the model printability. In our framework, the input model is first oriented in a way to minimize area that needs to be supported. Overhang points requiring the support are then detected. From these points, the support structures are iteratively built. Instead of column-like supports projected from the surface, we use tree-like structures that effectively support all overhangs. Branches are progressively merged until only one column is reached. This approach greatly reduces the length of the supports, while considering properties of the printer and object printability. Our solution is optimized for FDM printers, as they are the most accessible 3D printers on the market regarding price of the device and material.

We compared our solution to the supports generated by MakerBot® Makerware™, a 3D printing software supplied with MakerBot® 3D printers, and Autodesk® Meshmixer™ [Aut14] software that also provides support generation. Compared to the built-in Makerware™, our solution provides significant savings for both the material and printing time. On average, the printing of all objects took 29.4% less time and consumed 40.4% less material. The savings against Meshmixer™ software was less significant but on average, our solution was still capable of reducing the printing time by 11.8% and the material usage by 12.4%.

2. Previous work

Computer graphics models are mathematical representations of shapes and forms that are usually designed for direct visualization on the computer screen. Just very recently, when 3D printers became widely available, it has been noted that the 3D models may not be structurally sound, they may not be suitable for printing, and they may cause problems during or after fabrication. This problem has motivated many researchers. Most of them addressed the possible printability and structural problems of the 3D models before they are printed. Telea and Jalba [TJ11] used voxel-based analysis to detect unprintable parts, e.g. parts thinner than printer resolution that could result in disconnected parts. Stava et al. [SVB*12] presented a FEM-based framework to detect the most common structural problems. Part of the framework was automatic methods to relieve stress from the model through thickening, hollowing, or inserting struts into the model. An alternative to the costly FEM analysis has been proposed by Umetani and Schmidt [US13]. Their approach is based on detecting the geometric relationships between cross-sections of the object. The weakest cross-sections are found and, based on their orientation, printing direction is changed to increase the model strength. Recently Zhou et al. [ZPZ13] used a method to perform structural analysis based on the geometry and material properties only, searching for the worst possible loads likely causing damage to the model. Hildebrand et al. [HBA13] attempted to minimize direction bias and different tensile strength between distinct layers of the material by slicing the object into parts with optimal printing directions determined for each part. Prevost et al. [PWLSH13] introduced a model analysis to generate objects that are stable in a rest position.

Several attempts to reduce 3D printing volume through support optimization have been published. The simplest way to reduce the need for supports is to select the right object orientation on the printing tray, as addressed by Alexander et al. [AAD98]. The faces of the convex hull of the object represented the possible rest position to examine and the best position is selected based on the criteria of minimal support volume and the contact area. Recently, Vanek et al. showed how to pack objects into a 3D printing tray to speed up the printing and reduce support material volume [VGB*14]. Another way to reduce build volume is segmenting the model into multiple parts, as proposed by Luo et al. [LBRM12]. Possible cuts were found by using a beam search strategy with several optimization criteria such as part size, structural soundness, and visual impact. Connectors were generated on the cuts to make parts easier to assemble. In some cases, human interaction was needed to better select the cuts and lower visual impact on the model. A limitation of this approach was using the planar cuts only, without following the model features. Also, as the primary idea of this work was to make possible building objects larger than the printing tray, no support optimization has been employed.

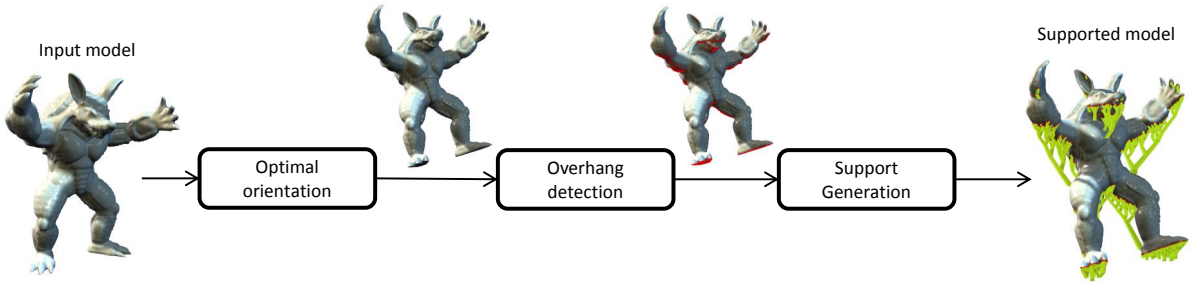

Figure 2: *System pipeline. The 3D model is first oriented into the position that requires the least amount of support. Points requiring support are then detected and the support structure is constructed by recursively connecting these points with the ground or the object itself. The result is a self-supporting and printable 3D model.*

The work of Wang et al. [WWY*13] addressed the problem of reducing the material usage in 3D printing more directly. The input model was converted into a skin-frame structure that is lightweight, while still maintaining the stiffness of the original object. Authors also showed basic optimization of the model supports by first detecting the overhang points and second generating simple, rod-like supports by connecting the overhanging points to the nearest point on the mesh or on the ground. A more advanced approach, similar to our solution, has been used in Autodesk®Meshmixer™ [Aut14] with tree-like supporting structures. The support generator requires some user interaction, it is not fully automatic, and it can fail in some cases. Moreover, as it is a part of the proprietary software, the exact description of the algorithm is not available. Our solution is fully automatic and achieves higher savings.

The construction of skeletal structures or *trusses* that are similar to our supporting structures has been studied in structural design. Several works on how to optimize the topology of trusses exist in the field of mechanical engineering (see the book of Bendsoe [Ben03]). However, the problem of optimal structures connection also arises in many other fields, for example circuit design [CW05] and wireless networks [MDJ*06]. Different criteria of optimality have been explored by Cheng [Che95], where the author proposes formulations for minimum weight, minimal stress, and no bucking constraints in the generated structure.

We use a pure geometrical approach in our work and the problem of minimizing the support structure is related to *the minimum Steiner tree problem* that has been addressed in many ways (see the surveys [GP68] and [HR92]). In particular, the support structure generation can be described as the Euclidean Steiner Minimal Tree problem (ESMT). This problem is at least NP-Hard. Several heuristics exist but most of them work only in 2D. The work of Toppur and Smith [TS05] is one of the few approaches in the 3D space that uses heuristics and gives a solution in $\mathcal{O}(n^2)$. Our solution uses a similar heuristic that is adapted to fulfill the extra constrains that arise in our setting.

3. Overview

Support structures are needed to print overhangs that are parts of the model that do not have support from below as shown in Figure 4. The most obvious support structures are struts perpendicular to the printing direction, a solution currently supported by 3D printing software, as shown by the red lines in Figure 4. However, this solution consumes a lot of material and is slow for printing. The 3D printers usually tolerate certain non-perpendicular angles for the support structures and the struts can be inserted under angles smaller than 90^o. Our task can be expressed as an optimization problem that minimizes the total length of the support structure while maintaining the ability to print overhangs.

The system pipeline is shown in Figure 2. The input is a 3D model represented by a boundary mesh. Further input parameters of our framework are 3D printer dependent and they include: minimum layer thickness the printer is able to produce and maximum surface angle the printer can print without errors (critical angle α_c on Figure 4).

In the first step, the object is oriented to a position that minimizes the area with overhangs by using the method of Alexander et al. [AAD98]. During the evaluation of the best position, parts of the model that need to be supported are detected. These parts are sampled (with sampling distance dependent on 3D printer capabilities) and points that require support (overhangs) are inserted into an ordered list. Orientation with the fewest overhang points is selected. The configuration with the smallest area that requires support does not need to lead to the shortest support structure. Figure 3 shows this dependency for the object from Figure 6. The object was rotated to 50 random positions and the support structure was constructed. The graph depicts the area requiring support and the actual support length. While using the sampling from all direction would lead better result, it would be also time consuming. Therefore, we use this simpler and faster heuristics.

The goal of the next step is to provide structures that minimize the total length of the resulting support structure. This

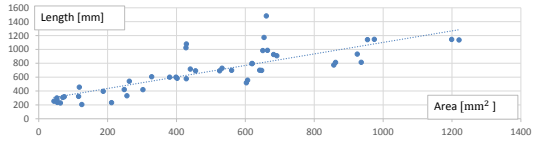

Figure 3: *A graph of the relation of the total support length and the support area for the object from Figure 6 rotated to 50 random positions.*

structure has a tree-like shape as it starts with the list of points that need to be supported (leaves) and gradually converges into one support (trunk) that connects to the ground or other parts of the mesh as shown in Figure 7. The structure is converted into a polygonal mesh that connects the original object in a way that is printable and easy to remove.

4. Overhang Detection

We can detect the following three types of overhangs.

- *Point overhang* is a point that is positioned lower than its neighboring points (local or global minimum).
- *Face overhang* is a triangular face. It is considered an overhang if the angle α between the plane defined by the face and the printing direction vector Y_P is higher than the critical angle α_c. Figure 4 shows two examples The face f_1 has $\alpha_1 < \alpha_c$ and it can be printed without support. However, face f_2 has the angle $\alpha_2 > \alpha_c$ and therefore it requires support.
- *Edge overhang* is defined similarly to face overhang. Edge normal is defined as the average of the normals from the two incident faces.

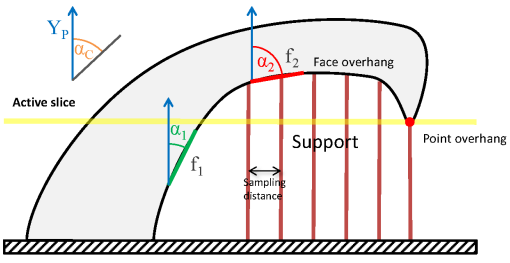

Figure 4: *Point and face overhangs. A point is considered an overhang if all its neighboring points are located higher. A face needs support if the angle between the face plane and the printing direction vector Y_P is higher than the critical angle α_C.*

The value of critical angle α_c is a crucial part of the optimization. The key observation is that 3D printers are capable of printing the faces with quite a large angle deviation from the printing direction vector. More specifically, most of the FDM printers are capable of printing faces deviating by up to 45^o from the printing direction vector. The exact value of the α_c varies from printer to printer and is not generally accessible. We have measured the α_c by a series of experiments printing the struts with different values of α_c and observing at which angles the struts started to collapse as shown in Figure 5.

Figure 5: *An example of the printed test setup used to find the value of α_c and the optimal length of supporting strut.*

The support is generated for points. Therefore, to support faces and edges we sample them by using uniform sampling. The sampling rate is directly related to the printer resolution. For example, if the resolution is 0.1 mm, it is safe to sample the overhang area with point distance equal to this distance. In practice, since the FDM printers are capable of stretching the melted plastic over the distance larger than their resolution, the sampling distance can be larger and we used 0.25 mm in our experiments. We use a quick hardware-oriented scanline rasterization algorithm to find sampling points on the triangles. The result of this step is the set of points P requiring support as shown in Figure 6.

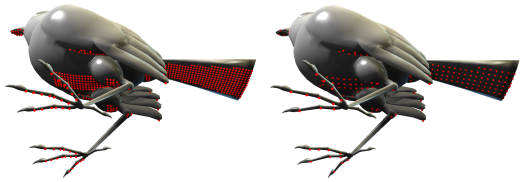

Figure 6: *Input 3D model with points requiring support sampled with different sampling distances.*

5. Support Structure Generation

We restrict our support structures to linear connections only. Let's define a vector $\vec{v}$ perpendicular to the printing direction; $\vec{v}$ is usually pointing up against gravity. Every point $p \in P$ should be supported by a structure that connects it with a point s. The location of this point is limited by the angle between the vectors $\vec{sp}$ and $\vec{v}$ that must be less or equal than the critical angle α_c. This space has a shape of a cone with its apex in p (see Figures 9). The set of valid points $s \in S$ is inside the cone such that the vector $\vec{sp}$ has an angle $\alpha \leq \alpha_c$

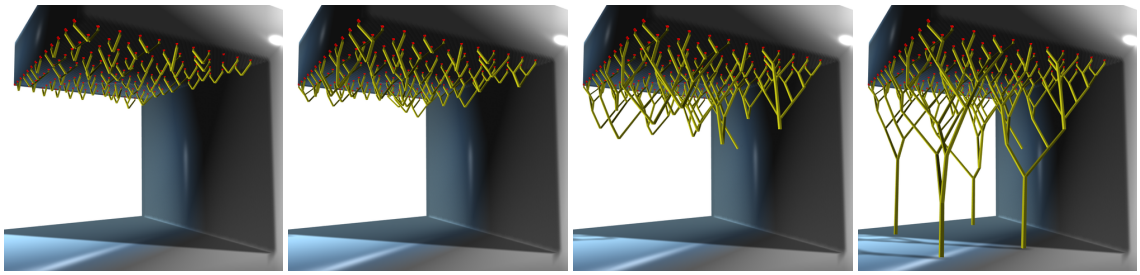

Figure 7: *Progressive building of the support structure. The strut diameter is dynamic; the first iteration is thinner than the last iteration as it is supporting less weight.*

with vector $\vec{v}$. We will refer to it as a *support cone* $\mathcal{C}$, as it defines the space where point s can be located such that the linear structure $\overline{ps}$ will support printing of point p.

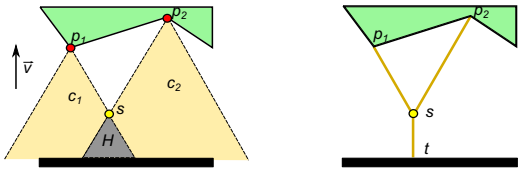

Figure 8: *Two points p_1 and p_2 requiring support in 2D and their supporting cones $\mathcal{C}_1$ and $\mathcal{C}_2$. The feasible space H to put a support point s that would support both points is in the intersection of the cones and the optimal place to put s is on the highest point of H. The resulting support structure for this case is on the right with the point t marking an intersection with the ground plane.*

Having two points p_1 and p_2 with their support cones $\mathcal{C}_1$ and $\mathcal{C}_2$ (see Figure 8 left) their intersection H includes all feasible places where we can put a support point s that would support both points p_1 and p_2. Let's also define t as the closest point on the ground to s (Figure 8 right). We have decided to use heuristic by selecting the closest intersection between two cones as the location of the point s (highest point of H on Figure 8) as it gives good results in minimizing the total length of the structure.

The union of all support cones $\mathcal{C}_i$ from all $p_i \in P$ defines the space where the supports for the whole object can be placed. This union will be searched for intersections between cones as they mark places where the support structures can be merged in the branching points.

The supports are built as a set of connected linear struts. We define a geometric graph $G = (V,E)$ whose set of vertices V is the union $V = P \cup S \cup T$, where:

1. P is a set of points on the mesh that need support,
2. S is a set of branching points where supports are merged together, and

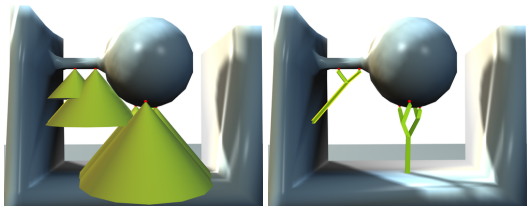

Figure 9: *Practical example of supporting cones generation. Support cones are placed with their origin at the overhang points and they represent space where the potential supports can be placed (left). By using the algorithm to find all cone to cone and cone to mesh intersections, supports are merged in the branching points, creating a tree-like structure (right).*

3. T is a set of points where support structure intersects with the model surface or the printing tray (see point t on Figure 8 right).

The optimization goal is to determine points in T and S that minimize the sum of the lengths of all edges $e \in E(G)$. This problem is related to the Euclidean Steiner Minimal Tree (ESMT) problem identified by Hwang and Richards [HR92]. The goal of ESMT is to find the tree of minimal length that makes the input set of points connected. In order to achieve this, it is permissible to add points that minimize the distance, and such points are called *Steiner points*. In our case, the geometric tree lies in 3D space and we have the additional constraint that the points in S (Steiner points) need to be inside the supporting cones $\mathcal{C}$ of the vertices that are adjacent to them. The complexity class of the ESMT in 3D Euclidean space is not known, but it should be at least as hard as is in the plane which is NP-complete, as shown by Toppur and Smith [TS05]. Therefore, the use of a heuristic to solve this kind of problem will be necessary.

Our solution to the problem consists of several steps and we use the plane sweeping algorithm. We know that for two points in V, the optimal point $s \in S$ supporting them lies at the intersection of their supporting cones (Figure 8). We group points by pairs and insert a support point s in this

place. Since we select a local minimum for every pair of points and we build our support tree iteratively from them, it is a greedy strategy.

The input of the algorithm is a list of overhang points P, sorted from top to bottom, and sorted list of cones $c \in \mathcal{C}$ placed at every overhang point $p \in P$ representing the space where the potential supports can be considered and where the intersections will be searched for. Then for each $p \in P$ and associated $c \in \mathcal{C}$ we perform the following steps.

1. Find the nearest intersection of c with the mesh m and the set of all other cones $\mathcal{C} \setminus c$. We compute cone to cone intersection with a procedure similar to the one described by [Ebe08]. Cone to mesh intersection is computed on the GPU by drawing the cone geometry with the mesh and reading the minimum depth value marking the intersection point.
2. Select the intersection s nearest to the point p and corresponding intersected cone c_i (cf. Figure 8 for illustration). If this intersection is further than m, remove p and c from the corresponding lists and continue to Step 1.
3. Insert s into the ordered list P so that s will become the new overhang point. At this point, two support lines (from p and from origin of c_i) merge at s into one.
4. Add cone in the place of s, remove p from P and c with c_i from $\mathcal{C}$
5. Get a new point p from P and repeat the procedure while P is not empty.

This algorithm progressively builds the supporting structure until all intersections that exist are with the mesh and the ground only (Figure 7). At each step, every point $p \in P$ is either preserved or grouped with the neighboring point. Therefore, as the size of P is reduced linearly, the algorithm has a complexity $\mathcal{O}(n)$. The most expensive part of the process is computing the intersection of a support cone with the mesh. The actual implementation uses the GPU and depth buffer to find the intersections (see Section 7).

6. Strut Design

Figure 10: *Tested extrusion profiles for the support structure.*

The output of the algorithm is a tree structure with points and connectivity information. To make this structure printable, a conversion of the supporting structure into a printable boundary mesh representation is necessary. The conversion is done by extruding the cross-section profile along the line connecting two points in the structure. The profile should be quickly printable while providing good support.

The profiles can be categorized depending on how many printing head movements are required to print one layer. Single movement means the support cross-section is a line or circle, two movements form a T-like shape, three movements can form a triangle, etc. To select the best profile, we conducted tests on the MakerBot® Replicator™ 2 printer and evaluated the profiles from Figure 10. From left to right, the number of printing head movements required to print the profile increases and we selected sample profiles constructed from one, two, and three lines. Profiles with four or more lines took too long to print.

We have printed various shapes by using the above-described profiles and tested their stability and their printing time. One- and two-line profiles showed unsatisfactory support for the model surface. Closed profiles (circle and triangle) were unsuitable because the 3D printer tends to fill the inside of the profile, resulting in a longer printing time. The most feasible was the N-shaped profile, as it requires only three movements of the printing head and, at the same time, it supports the entire area of a square. The thickness of the profile is equal to the printer resolution.

Appropriate diameter of the supporting strut depends on multiple factors such as strut length, tensile strength of the material, thickness of the deposited layer, movement speed of the printing head, among others. It would be possible to consider full structural analysis to progressively compute stress while the object is being built, but it would be computationally expensive. Therefore, we have decided to address the problem from the geometric point of view. We considered only the length and angle of the support and performed tests to find the relationship between the support strut diameter, length, and angle. Since it is problematic to directly estimate these parameters as they depend on the 3D printer and used material, we conducted experiments with different diameters of supports along with various strut lengths and angles. Figure 5 shows an example of such experiment. For each combination of settings we noted whether the support collapsed and what the quality of support structure was.

DIAMETER: 1mm						
length/angle	0	10	20	30	40	50
1cm	4	4	4	4	4	4
2cm	4	4	4	4	4	4
3cm	4	3	4	3	3	3
4cm	3	3	3	3	3	1
5cm	3	3	3	3	1	1
6cm	2	2	2	2	1	1

Table 1: *The stiffness experiment for the strut with* 1 *mm diameter. Quality of the supports was subjectively evaluated on the scale* $[0,4]$. *High values (4,3, green) mean strut was printed without problems whereas low values (2,1, yellow) indicate that strut was damaged or collapsed entirely.*

Table 1 shows the results on the color scale from green (support strut printed without any distortions) to yellow (strut was damaged and collapsed completely). We can see that diameter 1 mm is safe to use until lengths up to 3 cm with an angle no larger than 30^o. Diameter 1.5 mm can be used up to 6 cm in length and a 50^o angle. Exact expression of diameter from angle and length proved to be difficult, as more time-consuming prints with different diameters would be necessary. We did not test the struts with variable weights at the end because it would lead to a large number of test prints.Based on test results, we used this heuristic to determine the diameter:

$$d = k.l.\alpha, \quad (1)$$

where d is the strut diameter, α is the strut angle, and k is the length-to-angle coefficient. The value of k was estimated from test runs with three different diameters (0.5, 1.0, and 1.5 mm) on lengths from 1 to 6 cm and angles from 0^o to 50^o. In total, 108 combinations have been printed. Based on the test analysis, $k = 0.0015$ was giving satisfactory results.

The struts have variable diameters as they are computed independently for each strut end by using Eqn (1). Diameters are thinner at the top of the strut and greater at its end. The reason is the smooth transitions in branching point as the first iteration of supports is short and thin while the last iteration is much longer and needs to be thicker (Figure 7). Since the resulting structure has a tree-like shape, an alternative way to determine correct diameters could be by using an algorithm considering the relation of branch radii in biological trees [HBM03].

To allow easy detachment of the support from the rest of the model, a tip is added at its end as shown in Figure 11. Correct length of the tip depends again on the 3D printer and experiments were necessary to find the best value (we used the length of 1 mm).

Figure 11: *Tips are added at the ends of the support struts for easier removal from the model.*

7. Results

We have implemented our system in C++ with the use of OpenMesh library for mesh processing, Qt library for user interface, and OpenGL API to render the results. All tests have been performed on a PC equipped with Intel Core i7 920 2.67 GHz and nVidia Quadro K4000 with 4 GB VRAM.

The most computationally intensive part is finding the intersections of cones with the mesh. It has been implemented on the GPU by drawing cones and mesh into the depth buffer and extracting the lowest value that represents the closest intersection. The benefit of this implementation is that the supports are calculated in a matter of seconds

Our testing platform was a 3D printer MakerBot®Replicator™2 with PLA material, standard layer thickness 0.2 mm and capable of printing overhangs with a critical angle up to 50^o. This value was determined by testing, as the manufacturer does not provide this information. We used default (medium) quality settings for all prints.

We have compared our support generator with two available systems: Makerware™, the 3D printer software shipped with MakerBot®printers [Mak14] and Autodesk®Meshmixer™software [Aut14]. All software settings were kept to the default with the exception of Meshmixer™– in order to get comparable results, we set the same critical angle and support diameter we used in our application. Because our application uses a dynamic diameter while Meshmixer™uses a fixed one, we used a compromise value of 2.5 mm.

Eight triplets of models (24 in total) were printed. Some of them are shown in Figure 13. Table 2 summarizes statistics for all models showing the printing time, model weight, and percentage of the material and time saved when comparing our method against the others. Results show that in all cases, our solution achieved a significant improvement above the built-in Makerware™software. The average savings were 40.5% for material (ranging from 24.5% to 68.1%) and 29.4% for time (ranging from 13.9% to 49.6%). Figure 13 (rightmost column) shows the difference in the volume of removed support material.

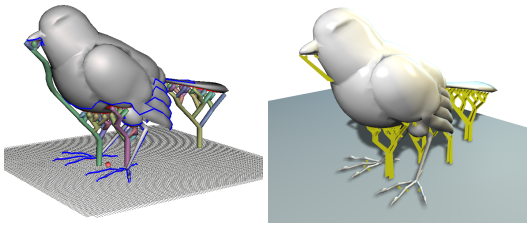

Figure 12: *Comparison of supports generated by Meshmixer (left) and our solution (right).*

Compared to the Meshmixer™software, the savings were not so significant but overall, our solution was able to save 11.8% more time (ranging from 2.4% to 22.5%) and 12.4% more material (ranging from 1.4% to 35.5%). We believe that the main contributor to the faster printing time of our solution are generally shorter supporting structures, as supports generated by Meshmixer™appear to be longer (see comparison on Figure 12). The N-shaped profile of the strut

	Makerware		Meshmixer		Clever supports					
Model	time (min)	weight(g)	time(min)	weight(g)	time(min)	weight(g)	saved time (Makerware)	saved time (Meshmixer)	saved material (Makerware)	saved material (Meshmixer)
Tree	355	37.96	231	18.78	179	12.11	49.58%	22.51%	68.10%	35.52%
Molecule	197	33.18	138	18.24	117	14.21	40.61%	15.22%	57.17%	22.09%
Armadillo	220	44.56	165	27.97	161	27.54	26.82%	2.42%	38.20%	1.54%
Arches	182	45.60	150	28.86	146	28.45	19.78%	2.67%	37.61%	1.42%
Bird*	93	20.75	N/A	N/A	80	14.35	13.98%	N/A	30.84%	N/A
Wheel	503	82.51	374	61.91	347	60.33	31.01%	7.22%	26.88%	2.55%
Dragon	220	39.96	210	34.01	167	30.17	24.09%	20.48%	24.50%	11.29%
* Bird model generated by Meshmixer was unprintable						AVERAGE:	29.41%	11.75%	40.47%	12.40%

Table 2: *Statistics of printed examples showing the printing time, total model weight with supports, and savings compared to the Makerware™ and Meshmixer™ software.*

also prints faster, consumes less material, and leaves less disturbing marks on the surface as compared to the round profile used by Meshmixer™.

Less support material also means faster cleaning of the object and a cleaner surface. Figure 13 shows that there is no visible difference in the model surface quality among different support generators after the supports have been detached from the model.

8. Conclusion

We have introduced an optimization framework attempting to minimize the supporting structures needed to 3D print objects containing overhangs. The object is first oriented into an optimal position and points requiring the support are detected. A tree-like structure supporting these points is constructed by merging the struts from initial points (leaves) into a single strut (stem) by using a greedy strategy. This support structure is fast to build and requires much less material to print than supports generated by other 3D printing software packages with support generators. We have been able to save almost one half of the support material as compared to 3D software shipped with the 3D printer while printing the model significantly faster.

Possibly the most serious limitation of our approach is that our support builder considers only the angle and length of the supporting strut. Our approach is geometry-based. As a progressive structural analysis of each printed layer would be time consuming, we do not provide any structural evaluation whether the support is able to withstand the weight of the supported part or shear forces during printing. It could be determined using software solution of [SVB*12], or with more extensive testing with different heights at the end of struts. Another limitation is that many parameters, such as the critical angle, sampling distance, and support diameter are closely connected to the selected 3D printing platform (we have used MakerBot® Replicator™ 2). Even though we were able to estimate these parameters by running various tests on the printer, it would be good to provide a more robust solution to this problem. Also, even though our greedy algorithm finds a good and fast solution, the question whether it is actually the minimal support structure remains to be answered.

We believe that most of the limitations could be addressed as future work. With added structural constrains and optimized structural analysis one would be able to compute stress and estimate if the strut will be able to support the overhang. A different approach would be to create solid support structure and progressively carve holes to lighten the structure while maintaining structural stability. Moreover, it could further improve the solution if we could find a fast strategy to find an optimal rotation rather than using the current minimal area criterion. Exact values for the critical angle, sampling distance and other parameters required for optimization could be obtained by a closer cooperation with 3D printer manufacturers.

References

[AAD98] ALEXANDER P., ALLEN S., DUTTA D.: Part orientation and build cost determination in layered manufacturing. *Computer-Aided Design 30*, 5 (1998), 343 – 356. 2, 3

[Aut14] AUTODESK®: Meshmixer™. http://www.meshmixer.com/, 2014. [accessed 10-4-14]. 2, 3, 7

[Ben03] BENDSOE M. P.: *Topology optimization: theory, methods and applications*. Springer, 2003. 3

[Che95] CHENG G.: Some aspects of truss topology optimization. *Structural optimization 10*, 3-4 (1995), 173–179. 3

[CW05] CHU C., WONG Y.-C.: Fast and accurate rectilinear steiner minimal tree algorithm for vlsi design. In *Proc. of the Intl. Symp. on Physical Design* (2005), ACM, pp. 28–35. 3

[Ebe08] EBERLY D.: Intersection of a line and a cone, Mar. 2008. URL: http://www.geometrictools.com/Documentation/IntersectionLineCone.pdf. 6

[GP68] GILBERT E., POLLAK H.: Steiner minimal trees. *SIAM Journal on Applied Mathematics 16*, 1 (1968), 1–29. 3

[HBA13] HILDEBRAND K., BICKEL B., ALEXA M.: Orthogonal slicing for additive manufacturing. *Computers & Graphics 37*, 6 (2013), 669 – 675. Shape Model. Intl. (SMI) Conf. 2

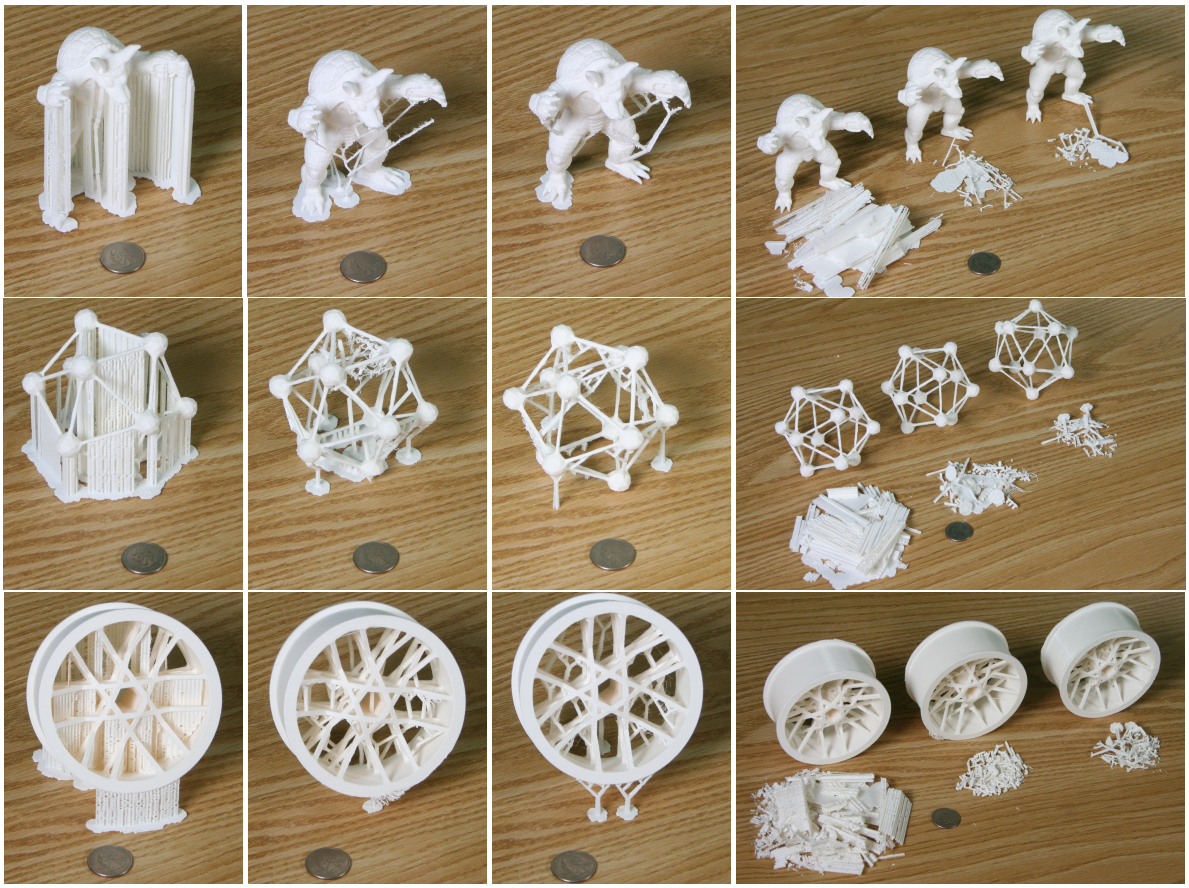

Figure 13: *Printed examples with the supports generated from different algorithms. First column (from the left) - Makerware™, second column - Meshmixer™, third column - our solution. The last column shows the differences in the amounts of support material after the objects have been cleaned. From left to right - Makerware™, Meshmixer™, and our solution.*

[HBM03] HART J. C., BAKER B., MICHAELRAJ J.: Structural simulation of tree growth and response. *The Visual Computer 19*, 2 (2003), 151–163. 7

[HR92] HWANG F. K., RICHARDS D. S.: Steiner tree problems. *Networks 22*, 1 (1992), 55–89. 3, 5

[LBRM12] LUO L., BARAN I., RUSINKIEWICZ S., MATUSIK W.: Chopper: partitioning models into 3d-printable parts. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 129:1–129:9. 2

[Mak14] MAKERBOT®: Makerware™. www.makerbot.com/makerware, 2014. [accessed 10-4-2014]. 7

[MDJ*06] MIN M., DU H., JIA X., HUANG C., HUANG S.-H., WU W.: Improving construction for connected dominating set with steiner tree in wireless sensor networks. *Journal of Global Optimization 35*, 1 (2006), 111–119. 3

[PWLSH13] PRÉVOST R., WHITING E., LEFEBVRE S., SORKINE-HORNUNG O.: Make it stand: balancing shapes for 3d fabrication. *ACM Trans. on Graph. 32*, 4 (2013), 81:1–81:10. 2

[SVB*12] STAVA O., VANEK J., BENES B., CARR N., MĚCH R.: Stress relief: improving structural strength of 3d printable objects. *ACM Trans. on Graph. 31*, 4 (2012), 48:1–48:11. 2, 8

[TJ11] TELEA A., JALBA A.: Voxel-based assessment of printability of 3d shapes. In *Proc. of the Intl. Conf. Mathematical morphology and its applications to image and signal processing* (2011), Springer-Verlag, pp. 393–404. 2

[TS05] TOPPUR B., SMITH J.: A sausage heuristic for steiner minimal trees in three-dimensional euclidean space. *Journal of Mathematical Modelling and Algorithms 4*, 2 (2005), 199–217. 3, 5

[US13] UMETANI N., SCHMIDT R.: Cross-sectional structual analysis for 3d printing optimization. *SIGG Asia Tech. Brief* (November 2013). 2

[VGB*14] VANEK J., GALICIA J. A. G., BENES B., MĚCH R., CARR N., ŠŤAVA O., MILLER G. S.: Packmerger: A 3d print volume optimizer. *Computer Graphics Forum* (2014), n/a–n/a. 2

[WWY*13] WANG W., WANG T. Y., YANG Z., LIU L., TONG X., TONG W., DENG J., CHEN F., LIU X.: Cost-effective printing of 3d objects with skin-frame structures. *ACM Trans. on Grap. 32*, 5 (2013). 3

[ZPZ13] ZHOU Q., PANETTA J., ZORIN D.: Worst-case structural analysis. *ACM Trans. on Graph. 32*, 4 (2013), 137:1–137:12. 2