Orientation Analysis of 3D Objects Toward Minimal Support Volume in 3D-printing

, Ben Ezair[a], Fady Massarwi[a], Gershon Elber[a]

[a]*Computer Science Dept. Technion Israel Institute of Technology, Haifa, Israel*

Abstract

In this paper, we examine how the support structure in 3D-printing can be optimized, by changing a model's orientation. Specifically, we explore the effect that the orientation of a printed object has on the volume of the needed support structure, directly below the model. We show that the volume of the support is a continuous but non-smooth function, with respect to the orientation angles. We continue by presenting an efficient algorithm, capable of running on a GPU, that computes the model's support volume for a given orientation. Then, this algorithm is used to seek an orientation with a minimal volume of support for constructing the model. Examples and experimental results are presented, showing that the minimum computed by our approach is, in many cases, optimal.

Keywords: Additive manufacturing, GPU computation, 3D-printing optimization

1. Introduction

The orientation in which a 3D model (denoted as M from now on) is constructed using a 3D-printer, affects the printing process in many ways. Properties such as mechanical strength, surface smoothness, overall printing time, and the amount of material needed for the support structure, all depend on the orientation in which M is constructed using 3D-printing [1] [2]. In this work, we concentrate on one such property: the volume of the support structure.

The support structure (denoted as S) is an additional printed structure, needed to support M (in our case directly from below, filling the entire vertical gap where the model overhangs) during the 3D-printing process. Without S, parts of M that have not yet achieved their full strength, may collapse during the printing process. While S is necessary to 3D-print certain models, the volume of S, $V(S)$, should be minimized as it requires additional printing time and material [3]. The main contributions of this work are:

1. Providing a better understanding of the problem of computing $V(S)$, given M.
2. Presenting an efficient algorithm to compute $V(S)$ for a given orientation, possibly using the GPU.
3. Developing a scheme by which the algorithm (presented in 2 and following the limitations portrayed in 1) can be employed to find the overall best orientation of M that minimizes $V(S)$. Our experiments indicate that in many cases the best solution found by our approach is, in fact, the optimal solution.

In Section 2, we give a short overview of related work. In Section 3, we present an efficient, possibly GPU based, approach to compute $V(S)$, for a given orientation of M. We show that $V(S)$ is a continuous but only piecewise smooth function of the orientation for a polygonal model M, and so the derivative of $V(S)$, with respect to orientation angles, cannot be assumed to exist for all orientations. Then, we present a derivatives-free algorithm that selects the best orientation for M, minimizing $V(S)$. In Section 4, we provide some experimental results, and finally, conclusions and possible future work are discussed in Section 5.

2. Related Work

An example of some of the early work related to finding the best printing (or build) orientation, is the work by Alexander et al. [3]. In [3], various orientation dependent properties, such as model height, support volume, and surface accuracy, are evaluated, and a cost comparison is used to find the best orientation. Other work similar to [3] was done by Frank et al. [4] that proposes an expert system tool that interacts with a user to select the best orientation, and Crawford et al. [5] that presents quantitative measures relating different aspects of part quality to orientation. In the work by Jibin et al. [6], the build orientation is optimized based on the considerations of the staircase effect (surface smoothness), support area (the total area of the downward-facing facets), and production time (model height). According to the authors of [6], a shortcoming of their method can be found in the long computing time. A recent overview of work related to finding the best printing orientation, can be found in the work of Taufik et al. [2].

Other related work concentrates on computing the effect that the orientation of M has on a single property, for example, in [7] a visual tool that presents a model's surface smoothness for a given orientation is used, allowing users to identify

the best orientation. The work by Gupta et al. [8] chooses a near optimal orientation based only on minimal build time (and minimal number of layers) for a shape deposition manufacturing (SDM) system. The optimal orientation in [8] is found by partitioning the unit sphere (representing all the candidate orientations) into smaller spherical polygons, identifying the best orientation within every spherical polygon, and assembling solutions from various polygons to find the final build orientation.

The most common single property examined is $V(S)$. The work by Allen et al. [9], computes an approximation of $V(S)$ for a given orientation, and also identifies a pool of good (small) $V(S)$ candidate orientations. $V(S)$ is approximated in [9] by sampling points on the surface of M and classifying them as supported or unsupported. Agarwal et al. [10] consider only convex models and use an approximation algorithm to compute an orientation, that minimizes the surface area of contact between M and the support structure. The work by McMains et al. [11] offers a fast GPU based algorithm for computing $V(S)$. The ability to run the algorithm on a GPU provides faster results than previous CPU based algorithms, while keeping the computation error under 1% for the cases presented in [11]. $V(S)$ computation times in [11], on a GPU are usually under a second, while the compared CPU based algorithm takes tens of seconds to complete.

Other works deal with aspects of the support volume other than orientation. For example, in the work by Hu et al. [12] the issue of support volume is sidestepped by decomposing the shape into (pyramidal) parts that can be printed individually without support material and then assembled together. In the work by Huang et al. [13] a reliable general support region is generated from part slices for a given orientation and the self-support area of the part (detected by offsetting the lower slicing contour), is excluded as much as possible to save support material.

This paper focuses on providing tools to compute $V(S)$ directly below M for given orientations, and also finds the best overall orientation that minimizes $V(S)$. The (possibly GPU based) algorithm we propose for computing $V(S)$ is similar to the one in [11]. In [11], $V(S)$ is also computed for a given orientation, using rendering. [11] fully computes, for each pixel, all polygons that cover (project to) the pixel and their distances from a base Z level, denoted the floor. The number of polygons covering each pixel is determined by the complexity of M. The maximum number of polygons covering a single pixel is referred to as the *depth complexity of M*. The distance information is then used to compute $V(S)$. They find this information by rendering M, layer by layer (in a process similar to Z-buffer peeling [14]): first, all the polygons closest to the screen plane in each pixel are peeled away. Next, the second closest polygons are peeled away and so on. The volume trapped between two adjacent layers is mapped to either inside or outside of M, with the volume outside of M assigned to $V(S)$. In contrast, the algorithm proposed here requires at most two renderings of M to compute $V(S)$, regardless of the depth complexity of M. We also provide a way to use the introduced algorithm to quickly find the global optimal orientation that minimizes $V(S)$.

3. Minimal Support Volume

Recall S is an additional printed structure that is needed to support M during printing. S is necessary to properly 3D-print certain models (some examples can be seen in [15], that explores a specific strategy to construct the support, and uses the approach given in [3] to select an orientation). The volume of S, $V(S)$, should be minimized by selecting the optimal build orientation, as S imposes additional costs in both extra material and printing time, costs that are proportional to $V(S)$ [3].

In Section 3.1, we lay down the theoretical foundation for our support volume computation algorithm. The algorithm itself, is presented in Section 3.2. Finally, in Section 3.3, we show how this algorithm can be used to select the overall best orientation.

3.1. Theoretical Background

Let M be a compact surface in R^3. In the following discussion, assume M is in a specific orientation, and the $+Z$ direction is considered up. We denote the lowest level (with the lowest value of z) in the model as Z_{min}. It's trivial to see that the optimal placement for the printing base surface (floor) in this orientation is at the height of Z_{min}: all higher values cause the floor to penetrate M, and all lower values add to $V(S)$ unnecessarily. The support structure needs to support every point in M not already supported by M. This means that every point in M must be connected to the floor by a vertical segment, made of points that are part of either S or M. The support S is formally defined as:

Definition 3.1.

$$\begin{aligned} S := \{\, (x,y,z) \mid z \geq Z_{min} \text{ and} \\ \exists z' > z \text{ so that } (x,y,z') \in M \text{ and} \\ (x,y,z) \notin M \,\}. \end{aligned} \tag{1}$$

Note that the mutual exclusion between M and S ($(x,y,z) \notin M$ in the definition of S), means that if M is defined as an open set, then S will be a closed set, and vice versa.

Definition 3.2. *The* top cover*, T, includes every point that is below some point $p \in M$ and is not below the floor (Z_{min}). Inclusion in T is defined as:*

$$\begin{aligned} T := \{\, (x,y,z) \mid z \geq Z_{min} \text{ and} \\ \exists z' \geq z \text{ so that } (x,y,z') \in M \,\}. \end{aligned} \tag{2}$$

Lemma 3.1. *For every point $p \in T$ either $p \in M$ or $p \in S$.*

Proof. This is a direct result of the definitions of the three sets. Clearly $S \subset T$, and $M \subset T$. On the other hand $T \subset (S \cup M)$. This leads to $T = (S \cup M)$, and since $S \cap M = \varnothing$, a point in T is either in M or in S. □

Corollary 3.2. *$S = T - M$ and $V(S) = V(T) - V(M)$.*

Because $V(M)$ is invariant to rotation, if we compute $V(T)$, V(S) can be derived with ease. See also Figure 1.

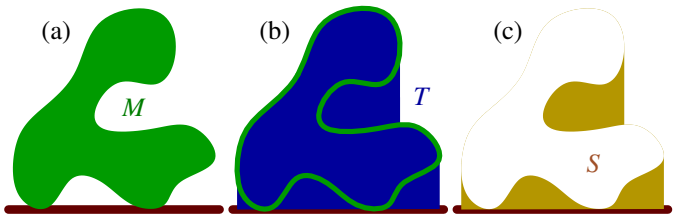


Figure 1: The Model M ((a), in green), has the top cover, T ((b), in blue). By subtracting M from T, S ((c), in yellow) is computed.

3.2. Support Volume Computation

We propose an algorithm that will take advantage of the observation made by Lemma 3.1 and Corollary 3.2, and the rendering abilities of GPU's to quickly compute $V(S)$. The idea behind the proposed algorithm is to map the depth value normally computed by a GPU, to the distance from some real plane (Z_{min}). Then, when the two, top and bottom, views of M are rendered with orthogonal projection, we will have the results for the first (nearest/lowest) and last (farthest/highest) Z level rendered in each pixel and can compute $V(T)$. The Z level of the lowest point in M (overall) can be set as the floor, Z_{min} of the printing surface. We then compute the volumes trapped between the highest Z level in each pixel and the level of the floor, and sum these volumes, only to get the value of $V(T)$. According to Corollary 3.2, in addition to $V(S)$ the extra volume enclosed in $V(T)$ is the internal volume of M, $V(M)$, that is invariant for all orientations and can be calculated in linear time [16]. By computing $V(T)$ using only two (top and bottom) simple Z-buffer renderings, we can derive $V(S) = V(T) - V(M)$, as opposed to [11] that requires multiple renderings, depending on the depth complexity of M.

Algorithm 1 implements the proposed approach and computes $V(S)$ for a given orientation. The algorithm starts by setting up the rendering to the correct resolution and position M so as to best fit the rendered image. The model is then rendered twice: once from above and once from below. The rendering from below is used to compute the lowest point in the model, Z_{min}, that is set to be the level of the floor. The rendering from above is used to compute the top cover for each pixel. The volumes, per pixel, are summed (and corrected for the Z_{min} level) to get $V(T)$. Finally, $V(S)$ is obtained from $V(T)$ using Corollary 3.2 as $V(S) = V(T) - V(M)$.

We should note that a single render approach may also be used for a polygon model, as the height of the lowest vertex can be set as the floor (or Z_{min}) level. Depending on the costs of rendering and information exchange between CPU and GPU, a single render approach may be more efficient on certain hardware setups. Some further optimizations are made to Algorithm 1 in our implementation, and are discussed in Section 4.

3.3. Optimizing Orientation

In Section 3.2, we presented an efficient algorithm to compute $V(S)$ for a given orientation. We now use this algorithm to identify the best orientation that minimizes $V(S)$. $V(S)$ as a function of the orientation angles (θ, ϕ) is potentially a non-smooth function, and is always non-smooth for polygonal models. Figure 2 shows some graphs for which $V(S)$, as a function of a single orientation angle θ (ϕ is set to a general position), exhibits a non-smooth behavior. Further analysis of the lack of smoothness and the behavior of $V(S)$, for a polygonal model M, as a function of the orientation angles (θ, ϕ), can be found in Appendix A.

Realizing $V(S)$ is a non-smooth function, we divide the problem of finding the best orientation (that minimizes $V(S)$) into two sub problems:

1. The first step performs a uniform sampling of orientation angles, in angular parametric space, and computes $V(S)$ as a function of these sampled angles. This sampling of $V(S)$ gives us a rough estimate of the best orientation.

2. The second stage is used to find a locally optimal $V(S)$ using a derivative-free optimization algorithm. Herein, we are following [17]. The algorithm accepts as parameters the ranges of θ and ϕ to be examined, as well as our function to compute $V(S)$ for a given orientation (i.e. Section 3.2). The final parameter for the algorithm is an initial guess for the search process.

   The algorithm in [17] converges to local optimums, that are identified as accumulations of points and designated

Algorithm 1 ComputeTopCoverVolume

Input:

(1) 3D model M.

(2) Orientation of M, prescribed using (θ, ϕ), angles of a spherical coordinates system.

(3) x_{res}, y_{res} the rendering resolution to use.

Output:

(1) $V(S)$ for the orientation of M dictated by (θ, ϕ).

Algorithm:

```
Setup the resolution of the renderer to x_res, y_res;
Set the rendering projection to tightly enclose M;
B_fb := Rendering result of M from below in orientation (θ, φ);
Z_min := min(Zbuffer(B_fb));
Initialize Zbuffer(A_fb(1..x_res, 1..y_res)) := Z_min;
A_fb := Rendering result of M from above in orientation (θ, φ);
H_sum := 0;
AreaOfPixel := RenderingAreaXY/(x_res y_res);
for all x ∈ 1..x_res do
    for all y ∈ 1..y_res do
        H_sum := H_sum + (Zbuffer(A_fb(x, y)) − Z_min);
    endif
    end for
end for
Return (H_sum · AreaOfPixel) − V(M);
```

as stationary points of the algorithm. Hence, it is not necessarily the case that the best sample obtained in step 1, will yield the overall best result in step 2. Hence, instead of using a single starting point, we use the best n results found in the sampling stage 1, and run the optimization algorithm [17] on each of the best n results. n is a parameter set by the user.

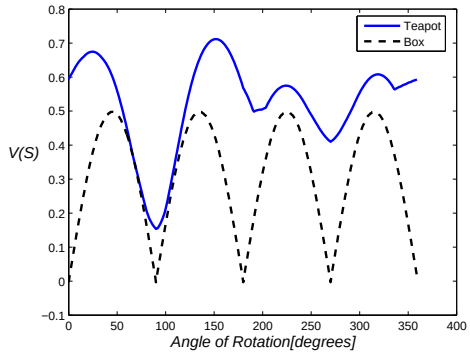


Figure 2: $V(S)$ for a cube (dashed black line) and a tessellated Utah Teapot model (shown in Figure 4 (b)) (solid blue line) in a general ϕ position, is a non-smooth function of the angle of rotation θ.

4. Results

In this section, we present some experimental results for both Algorithm 1, in Section 3.2, and the optimization scheme described in 3.3. The experiments were run on an Intel i7-3770 3.4 GHz, with 8 GB of RAM, and using a GeForce GT 630 GPU. Unless otherwise noted, the pixel resolution used is 1024^2, the sampling step size is five degrees, and n, the maximal number of samples fed to the optimizer, is 100. To achieve orientation (θ, ϕ), models are rotated (relative to their original orientation) $\theta \in [0, 360]$ degrees around the Z axis, and then $\phi \in [-90, 90]$ degrees around the Y axis.

As part of our implementation of Algorithm 1, we applied several small optimizations to the Algorithm:

- a single rendering approach was used which proved to be more efficient on our system (computing Z_{min} by simply traversing all vertices on the CPU).
- we use the mipmapping hardware of the GPU to hierarchically calculate the linear sum (H_{sum} in the algorithm). In our tests this mipmap hierarchical computation was found to be faster than the shader based hierarchical summing approach proposed in [11].
- since the same model M is going to be rendered from numerous view directions, we compute once a bounding sphere to M, properly position the bounding sphere in the rendering screen and rotate M around the center of the bounding sphere for all these numerous view directions

.

In Section 4.1, we explore the accuracy of the algorithm in computing $V(S)$. In Section 4.2, we examine the running times of the algorithm. Finally, in Section 4.3, we present some results for different models.

4.1. Accuracy

In the following results of this section, accuracy (the error) is expressed as the difference between the theoretical $V(S)$ and the computed $V(S)$, normalized as a percentage out of the theoretical $V(T)$ (that is, the total printed volume).

Figure 3 shows the accuracy of the computation of $V(S)$ for two simple freeform shapes, for which $V(T)$ and $V(S)$ can be determined analytically, as a function of the quality of the polygonal tessellation used. Figure 3 shows that for a sphere the deviation, from the theoretical $V(S) = \frac{1}{3}\pi r^3$ and $V(T) = \frac{5}{3}\pi r^3$ values, decreases as the tessellation quality (and the number of triangles in the model) increases, as expected. Figure 4 (a) presents a model of a cube (with a side length of 4) with four quarters of a unit sphere removed from it. $V(S)$ is calculated for an orientation in which two of the quarter spheres are on the downward facing face. Figure 3 shows again, for the model in Figure 4 (a), that the deviation from the theoretical $V(S) = \frac{2}{3}\pi r^3$, $V(T) = 64 - \frac{2}{3}\pi r^3$ decreases as the tessellation quality increases. Note the error in this case does not drop to zero because of aliasing errors explored in the coming examples.

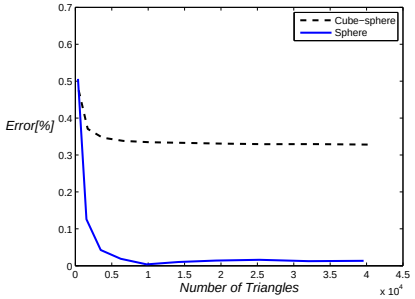


Figure 3: Accuracy of the computation of $V(S)$ as a function of quality of the polygonal tessellation, for a unit sphere model (blue solid line), and the model of a cube with four spherical quarters removed from it (black dashed line), shown in Figure 4 (a) (oriented so one of the faces with the removed spherical parts is parallel to the XY plane).

Figure 5 shows the computed $V(S)$ for a cube with one face parallel to the XY plane, rotated around the Z axis. The theoretical value of $V(S)$ for the cube is clearly zero, but inaccuracies due to aliasing errors cause an imprecise computation of $V(T)$ when the cube's edges align with the rows, columns, or diagonals, of the rendered image. Overall, the aliasing error remains small (far less than 1% of the volume of the cube).

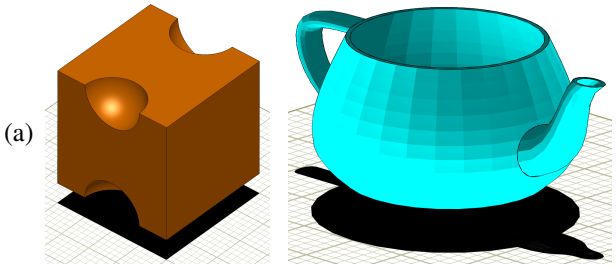


Figure 4: (a) A test model M, a cube with four spherical quarters removed from it. (b) A test model of a Utah teapot (with 3550 polygons). Both models are presented in the optimal orientation, that minimizes $V(S)$.

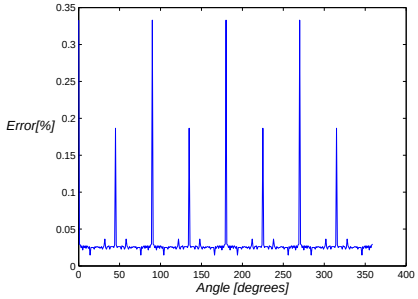


Figure 5: Computation accuracy of $V(S)$ for a cube (with one of its faces parallel to the XY plane) as it is rotated around the Z axis.

Figure 6 shows the computed $V(S)$ of a cube using different (pixel) resolutions of the rendered image. Again, the expected theoretical value of $V(S)$ is zero, but just as in the previous example, inaccuracies due to aliasing errors cause an inexact result to be computed. The results in Figure 6 show the expected behavior of randomized error values that diminish as the resolution increases.

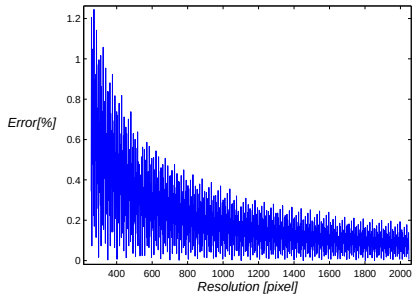


Figure 6: Computation accuracy of $V(S)$ for a cube (with one of its faces parallel to the XY plane) using different (square) pixel resolutions, of the rendered image.

In summary, our experiments concerning accuracy lead us to believe one can achieve a sub-percent accuracy, even for cubes or other shapes that exacerbate aliasing errors, using a GPU based approach, and a resolution of 1024^2.

4.2. Computation Time

Figure 7 shows the average time needed to compute $V(S)$ in a single orientation, for a model of a cube, using different (pixel) resolutions. The tested resolutions are all powers of 2 in order to take advantage of the mipmapping hardware that can substantially accelerate the computation. As expected, computation time increases as the resolution increases, but remains below 0.025 seconds for all tested resolutions, and is about 2 milli-seconds for 1024^2.

Figure 8 shows the typical time needed to compute $V(S)$ in a single orientation, for sphere models using different levels of tessellation quality (number of triangles). As expected computation time increases as the number of triangles increases, but in a sub-linear rate as the number of polygons increase.

Figure 9 shows the time needed for the sampling step, and the time needed to find the optimal result (using the optimization algorithm) for the model of a Utah teapot (the model in Figure 4 (b)). The maximum n used in our tests for the optimization step (as described in Section 3.3) is $n = 100$. For sampling steps that produce less than 100 samples all the samples from step 1 are used in the optimization step. Note that at roughly a sampling step of 25 degrees, the total number of produced samples is 100, which is why there's a distinct drop in the optimization time after that point, having less than 100 points to optimize. The $n = 100$ optimizations, are faster for less than 25 degrees than for 25 degrees, due to the fact that the optimization algorithm converges faster when the starting locations are closer to the solution.

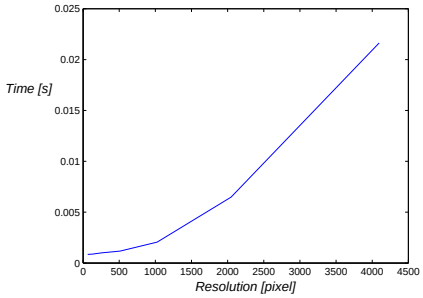


Figure 7: $V(S)$ computation time for a model of cube using different (square) pixel resolutions.

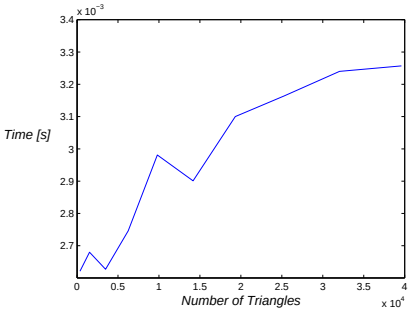


Figure 8: $V(S)$ computation time for spheres approximated using different numbers of triangles.

Figure 10 shows the optimal support volume $V(S)$ for the Utah teapot (Figure 4 (b)), achieved by just sampling, and that achieved by using the optimization algorithm. The figure shows that even without further optimization, a reasonably fine sampling can give fairly accurate results, in a reasonable amount of time. On the other hand, the optimization algorithm can find the optimal result ($V(S) = 0.1533$) even if the sampling resolution is quite coarse. For the Utah teapot model, the optimization algorithm only fails to find the optimal result when the sampling resolution is over 120 degrees. Figure 4 (b) presents the teapot in its optimal orientation.

Similar experiments performed for five other test models (see Figure 11) produce similar results: given a reasonably fine step size (sampling resolution), the sampling step gives results that are close to the optimum, while the optimization algorithm finds the optimal result even when a relatively coarse step size is used. In all our experiments, a sampling step of five degrees gave us a difference of less than ten percent between the best

optimization result and the best sampled result.

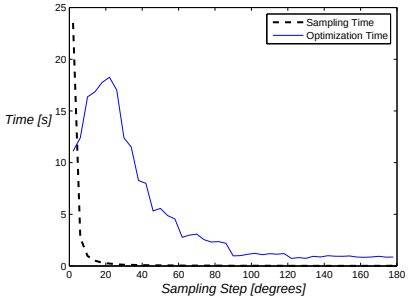


Figure 9: $V(S)$ sampling and optimization times for a Utah teapot (3162 triangles) for various sampling step sizes.

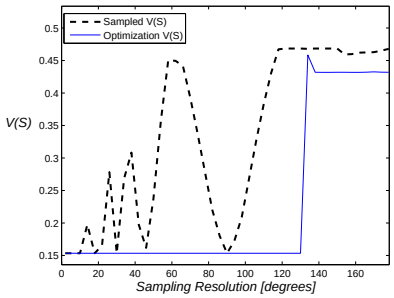
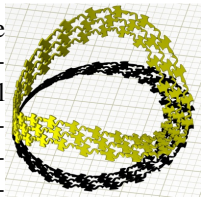


Figure 10: Optimal $V(S)$ achieved by sampling and by the optimization algorithm for different sampling step sizes, for a Utah teapot (3162 triangles).

4.3. Various Models

In this section, we present the experimental results of our algorithm for several complex models that are presented in Figure 11. Table 1 shows the results of the computation for these models. The sampling resolution used for these results is 5 degrees for both θ and ϕ (having $72 \times 36 = 2592$ samples), for optimization $n = 100$, and the pixel resolution used is 1024^2. The results in Table 1 are consistent with previous results (in Section 4.2): a reasonable sampling resolution provides quick and fairly accurate results, while the optimization algorithm is slower but can be used to achieve the optimal result. Figure 12 shows the five models in their optimal printing orientations. The columns in Table 1 are:

1. Model - name of the model, and a reference to its picture.
2. Triangles - the number of triangles in the model.
3. $V(M)$ - the computed volume of the model.
4. Sampling $V(S)$ - the best $V(S)$ value found by sampling.
5. Optimization $V(S)$ - the best $V(S)$ value found by the optimization algorithm.
6. Sample Time - the total time taken by the sampling stage.
7. Optimization Time - the total time taken by the optimization stage.



Of special interest is the result of the Moebius strip. The orientation that intuitively one can assume will yield minimal $V(S)$ is probably when the strip is in contact with the floor at two almost opposite locations (i.e. see image on the right). However, this intuition will result in a $V(S)$ that is roughly 50% larger than the optimum in Figure 12 (e).

5. Conclusions

This work presented tools to aid in the selection of the optimal orientation, for 3D-printing. We introduced algorithms to compute, and then identify the best printing orientation, based on the minimization of $V(S)$. We hope that future work will leverage this tool to formulate an overall printing orientation selection strategy. Such a strategy will have to weigh the optimization of $V(S)$ against other objectives, and choose the best printing orientation based on the overall objectives of a specific application, and the 3D-printing process.

Because $V(S)$ is a non smooth function, typically, and because derivative-free algorithms, like [17], are unlikely to yield a global optimum, one possible avenue for further research is the question of establishing bounds on the variations of $V(S)$ as a function of changes in θ and ϕ. Specifically, it may be possible to derive a bound, L_S, on the variation of $V(S)$ with respect to θ and ϕ:

$$|V(S)(\theta + \epsilon, \phi) - V(S)(\theta, \phi)| < L_s\epsilon,$$

and

$$|V(S)(\theta, \phi + \delta) - V(S)(\theta, \phi)| < L_s\delta.$$

Knowing the angular sampling step, L_S can then be used to bound the difference between the best result found using sampling, and the overall expected optimal result. Such a bound will allow one to produce an approximation algorithm for the optimal $V(S)$ possibly without an optimization step, and allow one to set the necessary angular sampling step to achieve a needed accuracy.

Another avenue for further research is the question of support that is not necessarily directly below M. An example of such a support structure is discussed in [15], with the clear potential advantage of having a smaller $V(S)$.

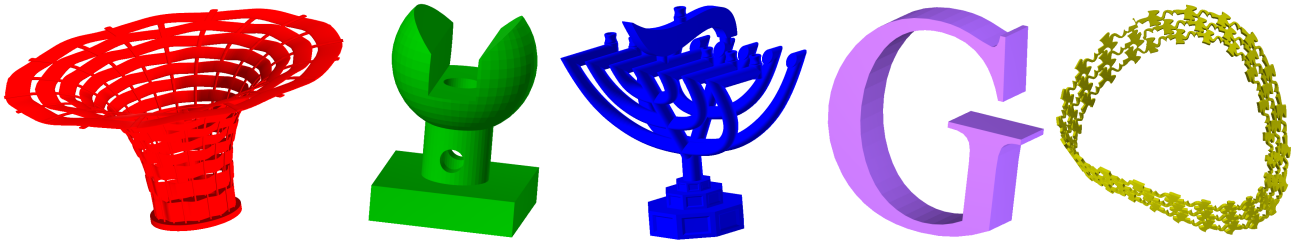
Finally, we have used a somewhat naive scheme to sample the angular space of orientations, by using a uniform sampling in the (θ, ϕ) angular parameter space. Clearly, one can exploit a more elaborate scheme of sampling the orientations uniformly in Euclidean space, over S^2.

6. Acknowledgments

This work was supported in part by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement PIAP-GA-2011-286426, and was supported in part by the ISRAEL SCIENCE FOUNDATION (grant No.278/13).

Model (Figure 11)	Triangles	$V(M)$	Sampling $V(S)$	Optimization $V(S)$	Sampling Time[s]	Optimization Time[s]
Wicker (a)	209401	0.007	0.114	0.103	8.145	31.445
Part (b)	2889	0.393	0.1303	0.1302	3.351	11.896
Menorah (c)	211004	0.081	0.1479	0.1478	8.337	34.573
G (d)	372	0.135	0.000	0.000	2.663	9.352
Moebius (e)	47571	0.006	0.0678	0.0675	2.949	13.992
Teapot (Figure 4 (b))	3550	0.096	0.1533	0.1533	3.802	12.819

Table 1: Results of computing the minimal $V(S)$ for several complex models of approximately unit size. Resolution is 1024^2 pixels, and the sampling step is five degrees.



(a) (b) (c) (d) (e)

Figure 11: (a) A wicker-like basket, (b) a mechanical part, (c) a model of dual views Israel's menorah emblem and the Technion logo, (d) an upper case G, (e) a Moebius strip, using ducks.

Appendix A. Smoothness and Behavior of $V(S)$

In this section, we discuss the smoothness of $V(S)$ for a polygonal model M as a function of the orientation. A polygonal model is a piecewise linear composition of a set of polygons, typically triangles, forming a closed 2-manifold. Given a triangle $\mathcal{T}$ having vertices $p_i = (x_i, y_i, z_i), i = 0, .., 2$, $V_\mathcal{T}$ - the volume of the truncated prism created between $\mathcal{T}$ and the XY plane:

$$V_\mathcal{T} = \frac{n_z(z_0 + z_1 + z_2)}{6}, \tag{A.1}$$

where $n = (n_x, n_y, n_z)$ is the unnormalized normal of $\mathcal{T}$ that can be obtained by $n = (p_1 - p_0) \times (p_2 - p_0)$. See also [18] for which $\frac{\|n\|}{2}$ is the area of $\mathcal{T}$. For simplicity, we study the behavior of $V_\mathcal{T}$ by rotating $\mathcal{T}$ around a single axis, the X axis. When rotating $\mathcal{T}$ around the X axis by angle θ, each z_i becomes:

$$z_i \rightarrow z_i \cos(\theta) + y_i \sin(\theta), \qquad i = 0, .., 2, \tag{A.2}$$

and n_z becomes:

$$n_z \rightarrow n_z \cos(\theta) + n_y \sin(\theta). \tag{A.3}$$

Substituting A.2 and A.3 in A.1 results in:

$$\begin{aligned} V_\mathcal{T}(\theta) = &\frac{n_z \cos(\theta) + n_y \sin(\theta)}{6} \\ &\Big((z_0 + z_1 + z_2)\cos(\theta) + \\ &(y_0 + y_1 + y_2)\sin(\theta)\Big), \end{aligned} \tag{A.4}$$

and after rearranging,

$$\begin{aligned} V_\mathcal{T}(\theta) = \frac{1}{6}\Big(&n_z(z_0 + z_1 + z_2)\cos^2(\theta) + \\ &(n_z(y_0 + y_1 + y_2) + \\ &n_y(z_0 + z_1 + z_2))\sin(\theta)\cos(\theta) + \\ &n_y(y_0 + y_1 + y_2)\sin^2(\theta)\Big). \end{aligned} \tag{A.5}$$

We show that the volume of support of triangle $\mathcal{T}$, $\widetilde{V_\mathcal{T}}(\theta)$, is not smooth as a function of θ by considering the simple individual case where the initial $\mathcal{T}$ is parallel to the XY plane, with the normal pointing down in relation to the $+Z$ direction. The volume of support, $\widetilde{V_\mathcal{T}}(\theta)$, of $\mathcal{T}$ vanishes after $\pi/2$ rotation, when n becomes coplanar to the XY plane. $\widetilde{V_\mathcal{T}}(\theta)$ can be written as follows:

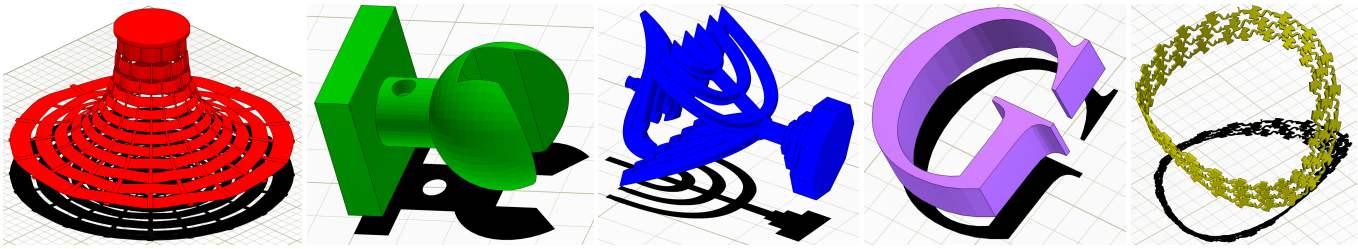
$$\widetilde{V_\mathcal{T}}(\theta) = \begin{cases} V_\mathcal{T}(\theta), & \pi/2 \geq \theta \geq 0, \\ 0, & \theta > \pi/2. \end{cases} \tag{A.6}$$

If $\mathcal{T}$ is parallel to the XY plane, $n_x = n_y = 0$, and $\widetilde{V_\mathcal{T}}(\theta)$ is reduced to:

$$\widetilde{V_\mathcal{T}}(\theta) = \begin{cases} \frac{n_z}{6}\Big((z_0 + z_1 + z_2)\cos^2(\theta) + \\ \quad (y_0 + y_1 + y_2)\sin(2\theta)/2\Big), & \pi/2 \geq \theta \geq 0, \\ 0, & \theta > \pi/2. \end{cases} \tag{A.7}$$

$\widetilde{V_\mathcal{T}}(\theta)$ is continuous at $\theta = \pi/2$. I.e $\widetilde{V_\mathcal{T}}(\theta)$ vanishes as the trigonometric functions vanish for $\theta \rightarrow \frac{\pi}{2}$, however, its derivative , $\frac{\partial \widetilde{V_\mathcal{T}}}{\partial \theta}$, does not exist there:

$$\lim_{\theta \rightarrow \pi/2^-} \frac{\partial \widetilde{V_\mathcal{T}}}{\partial \theta} = \frac{-n_z}{6}(y_0 + y_1 + y_2), \tag{A.8}$$



(a) (b) (c) (d) (e)

Figure 12: The models from Figure 11 presented in their computed optimal orientations. All models but (e) are shown "floating in the air" to better depict their optimal orientation. The result in (e) is a bit surprising as some portions of this model are substantially above the floor. Interestingly enough, in this optimal orientation, the cast shadow is of a very small area, explaining this somewhat surprising orientation.

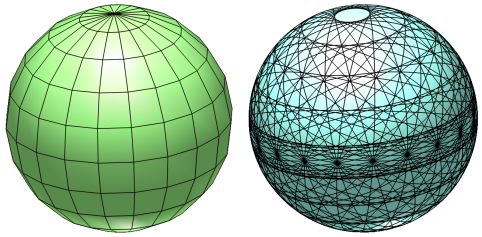
while

$$\lim_{\theta \to \pi/2^+} \frac{\partial \widetilde{V_{\mathcal{T}}}}{\partial \theta} = 0. \tag{A.9}$$

Hence, we can conclude the following:

Corollary Appendix A.1. *$V(S)$ of M is non-smooth at every orientation some polygon in M becomes vertical (or its normal becomes horizontal)*

Consider the Gaussian sphere S^2 and consider the great circle $C_{\mathcal{T}} \subset S^2$ orthogonal to normal $n_{\mathcal{T}}$ of some triangle $\mathcal{T} \in M$. Any view direction $v \in C_{\mathcal{T}}$ is a non-smooth transition in the support volume of $\mathcal{T}$. A plot of all great circles $C_{\mathcal{T}}$, for all $\mathcal{T} \in M$, will delineate the smooth regions of $V(S)$ in S^2. The order of the number of intersections of k great circles in S^2 is $O(k^2)$ and indeed the complexity of this arrangement can grow up rapidly. Figure A.13 show one simple example.



(a) (b)

Figure A.13: A simple model of a sphere approximation using 200 polygons (a) yields a fairly complex arrangement of smooth regions of $V(S)$, on the Gaussian sphere, on S^2 (b).

References

[1] Ahn SH, Montero M, Odell D, Roundy S, Wright PK. Anisotropic material properties of fused deposition modeling abs. Rapid Prototyping Journal 2002;8(4):248–57.

[2] Taufik M, Jain PK. Role of build orientation in layered manufacturing: a review. International Journal of Manufacturing Technology and Management 2013;27(1):47–73.

[3] Alexander P, Allen S, Dutta D. Part orientation and build cost determination in layered manufacturing. Computer-Aided Design 1998;30(5):343–56.

[4] Frank D, Fadel G. Expert system-based selection of the preferred direction of build for rapid prototyping processes. Journal of Intelligent Manufacturing 1995;6(5):339–45.

[5] Crawford DCTRH. Optimizing part quality with orientation. In: Solid Freeform Fabrication Symposium Proceedings. Center for Materials Science and Engineering, Mechanical Engineering Department and Chemical Engineering Department, the University of Texas at Austin; 1995, p. 362–8.

[6] Jibin Z. Determination of optimal build orientation based on satisfactory degree theory for rpt. In: Computer Aided Design and Computer Graphics, 2005. Ninth International Conference on. IEEE; 2005, p. 6–21.

[7] Kattethota G, Henderson M. A visual tool to improve layered manufacturing part quality. In: Proceedings of the Solid Freeform Fabrication Symposium, Austin, TX. 1998, p. 327–34.

[8] Gupta SK, Tian Q, Weiss L. Finding near-optimal build orientations for shape deposition manufacturing. In: Machining Impossible Shapes. Springer; 1999, p. 208–16.

[9] Allen S, Dutta D. On the computation of part orientation using support structures in layered manufacturing. In: Solid Freeform Fabrication Symposium 1994. DTIC Document; 1994, p. 259–69.

[10] Agarwal PK, Desikan PK. Approximation algorithms for layered manufacturing. In: SODA. 2000, p. 528–37.

[11] Khardekar R, McMains S. Fast layered manufacturing support volume computation on gpus. In: ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers; 2006, p. 993–1002.

[12] Hu R, Li H, Zhang H, Cohen-Or D. Approximate pyramidal shape decomposition. ACM Transactions on Graphics (TOG) 2014;33(6):213–25.

[13] Huang P, Wang CC, Chen Y. Algorithms for layered manufacturing in image space. In: Advances in Computers and Information in Engineering Research, Volume 1. ASME Press; 2014, p. 3–35.

[14] Everitt C. Interactive order-independent transparency. White paper, nVIDIA 2001;2(6):7–17.

[15] Vanek J, Galicia JA, Benes B. Clever support: Efficient support structure generation for digital fabrication. In: Computer Graphics Forum; vol. 33. Wiley Online Library; 2014, p. 117–25.

[16] Hughes SW, D'arcy T, Maxwell DJ, Saunders J, Ruff C, Chiu W, et al. Application of a new discreet form of gauss' theorem for measuring volume. Physics in Medicine and Biology 1996;41(9):1809–24.

[17] Lucidi S, Sciandrone M. A derivative-free algorithm for bound constrained optimization. Computational Optimization and applications 2002;21(2):119–42.

[18] Klamkin MS. On the volume of a class of truncated prisms and some related centroid problems. Mathematics Magazine 1968;:175–81.