Heuristics for Estimating Contact Area of Supports in Layered Manufacturing

IVAYIO ILINKIN
Rhodes College
RAVI JANARDAN
University of Minnesota
MICHIEL SMID
Carleton University
ERIC JOHNSON
University of Minnesota
PAUL CASTILLO
Lawrence Livermore National Laboratory
and
JÖRG SCHWERDT
Softwarebüro Bubel GmbH

Layered manufacturing is a technology that allows physical prototypes of three-dimensional(3D) models to be built directly from their digital representation, as a stack of two-dimensional(2D) layers. A key design problem here is the choice of a suitable direction in which the digital model should be oriented and built so as to minimize the area of contact between the prototype and temporary support structures that are generated during the build. Devising an efficient algorithm

Authors' addresses: Ivayio Ilinkin, Department of Mathematics and Computer Science, Rhodes College, Memphis, TN 38112; email: ilinkin@rhodes.edu. Research supported, in part, by NSF grant CCR–9712226. Ravi Janardan and Eric Johnson, Department of Computer Science & Engineering, University of Minnesota, Minneapolis, MN 55455; email: janardan@cs.umn.edu, erj@visi.com. Research supported, in part, by NSF grants CCR-9712226 and CCF-0514950. This effort was also sponsored, in part, by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAD19-01-2-0014, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred; Michiel Smid, School of Computer Science, Carleton University, Ottawa, Canada, K1S 5B6; email: michiel@scs.carleton.ca. Research supported by NSERC. Research supported, in part, by NSF grants CCR-9712226. Paul Castillo, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551; email: castillo17@llnl.gov. Jörg Schwerdt, Softwarebüro Bubel GmbH, 66459 Kirkel, Germany; email: jschwerdt@swbb.de.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2006 ACM 1084-6654/2006/0001-ART1.6 $5.00 DOI 10.1145/1187436.1210589 http://doi.acm.org 10.1145/1187436.1210589

for computing such a direction has remained a difficult problem for quite some time. In this paper, a suite of efficient and practical heuristics is presented for estimating the minimum contact area. Also given is a technique for evaluating the quality of the estimate provided by any heuristic, which does not require knowledge of the (unknown and hard-to-compute) optimal solution; instead, it provides an indirect upper bound on the quality of the estimate via two relatively easy-to-compute quantities. The algorithms are based on various techniques from computational geometry, such as ray-shooting, convex hulls, boolean operations on polygons, and spherical arrangements, and have been implemented and tested. Experimental results on a wide range of real-world models show that the heuristics perform quite well in practice.

Categories and Subject Descriptors: E.1 [**Data Structures**]; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; I.6 [**Computer-Aided Engineering**]

General Terms: Algorithms, Experimentation, Measurement, Performance, Theory

Additional Key Words and Phrases: Computational geometry, algorithm implementation and testing

1. INTRODUCTION

Layered manufacturing (LM) is a fast-growing technology with significant impact on the efficiency of the design process in a broad range of industries [Jacobs 1992; Chua et al. 2003]. LM offers a flexible and cost-effective alternative to traditional methods used in the design phase of physical prototypes. Current LM technology produces high-quality prototypes with added color in a matter of hours and at low cost. The prototypes can be inspected for flaws and, if necessary, the design can be modified and the process repeated until the final design has reached the desired quality.

Stereolithography is a widely used LM process. In essence, the stereolithography apparatus (SLA) consists of a vat of light-sensitive liquid resin, a platform, and a laser (see Figure 1). The input to the process (and to virtually all other LM processes) is a surface triangulation of the digital model in the industry-standard STL format. The model is oriented suitably and sliced into horizontal two-dimensional(2D) layers, which are then sent over a network to the SLA. The laser traces out the contour of each layer (a polygon) and then scans the interior in a zig-zag pattern. The exposure to the laser causes the scanned portion of the liquid to harden and form the physical layer. The platform is then lowered by an amount equal to the layer thickness (typically a few thousandths of an inch) and the next layer is then built on top of the previous one; thus, the three-dimensional(3D) prototype is realized eventually as a vertical stack of 2D layers. Ideally each new layer should rest completely on top of the previous one, so that the prototype is self-supporting during the build phase. Unfortunately, the complex shape of real-world prototypes often prevents them from being self-supporting (in any orientation). Therefore, during a preprocessing step, the model is analyzed and additional structures called *supports* are created and merged with the description of the model. Supports are built simultaneously with the prototype and later removed in a postprocessing step.

The choice of orientation can critically impact the efficiency of the build process and the surface quality of the physical prototype. Several competing criteria need to be addressed when choosing an optimal orientation. For example,

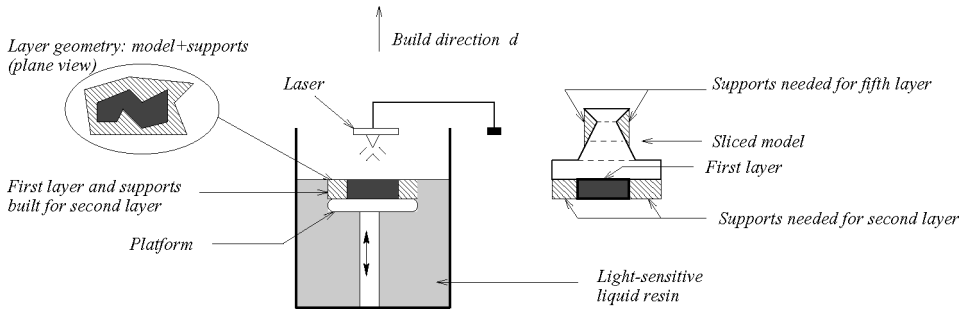


Fig. 1. The stereolithography apparatus, along with the sliced digital model and support structures.

an optimal orientation that minimizes the amount of support structures will, in general, lead to faster build times. Similarly, an orientation that minimizes the contact area (the area of that portion of the surface of the prototype that is in contact with supports) would help minimize damage to the surface of the prototype during support removal.

The problem of finding a suitable orientation can be translated into purely geometric terms and this has led to a considerable amount of research in recent years. Asberg et al. [1997] (see also [Bose 1995]) describe efficient algorithms to decide if a given model can be built without supports using Stereolithography. Majhi et al. [1999b] give algorithms to minimize the volume of supports and contact area for convex polyhedra. (see also [Majhi 1998].) Schwerdt et al. [2002] show how to choose a build direction, that protects prescribed facets from being damaged by supports. Agarwal and Desikan [2000] give an efficient algorithm to compute a build direction, which approximates the minimum contact area for a convex polyhedron. They also show that for a nonconvex polyhedron the set of directions for which the total area of the facets in contact with supports is minimum has $\Omega(n^4)$ connected components. (Note that this problem is not quite the same as minimizing the total contact area, as it considers the entire area of a facet in contact with supports regardless of area actually in contact; nevertheless, the $\Omega(n^4)$ lower bound hints at the potential difficulty associated with minimizing the contact area and provides further motivation for the heuristics we consider here.) Johnson [1999] shows how to compute support descriptions efficiently for a given build direction.

Unfortunately, very few results are available for the problem of optimizing support requirements for nonconvex polyhedra. Majhi et al. [1999a] give support optimization algorithms for nonconvex polygons in the case of 2D stereolithography. An exact algorithm to minimize contact area for polyhedral models is presented in [Schwerdt 2001], but its high running time precludes its use in practice; specifically, the running time is $O(n^6 q(n))$, where $q(n)$ is the time to solve a certain nonlinear optimization problem on the unit sphere. Allen and Dutta [1995] give heuristics for minimizing support contact area for nonconvex polyhedra. Their approach restricts candidate build directions to the (discrete) set of facet normals of the convex hull of the model and, furthermore,

considers only those facets whose areas are relatively large (so that the part is stable).

1.1 Contributions

In this paper, we make further progress on the contact area problem for polyhedral models. Specifically, we provide a suite of efficient and practical heuristics for estimating support contact area (Sections 4–6). These heuristics are based on various techniques from computational geometry, such as ray shooting, convex hulls, boolean operations on polygons, spherical algorithms, etc., and make use of the CGAL Computational Geometry Algorithms Library and LEDA [Mehlhorn and Naher 1999] libraries in the software implementation. We also give a method for evaluating the quality of the estimate provided by any heuristic, which does not require knowledge of the (unknown and hard-to-compute) optimal solution; instead, it provides an indirect upper bound on the quality of the estimate via relatively easy-to-compute quantities (Section 3). Finally, we present an extensive set of experimental results on real-world STL models that show that, in practice, the heuristics perform quite well (Section 7).

2. PRELIMINARIES

We denote by $\mathcal{P}$ the polyhedron of interest and by n the number of facets in $\mathcal{P}$. We assume that the facets of $\mathcal{P}$ are triangles and that its boundary is represented in some standard form, such as, for instance, a doubly connected edge list [de Berg et al. 1997] or a winged-edge structure [Baumgart 1975]. (If necessary, such a representation can be computed easily from the standard STL representation of $\mathcal{P}$ [McMains and Séquin 1999].) Let $\mathbf{d}$ be a given *build direction* (a unit-vector); we assume, w.l.o.g., that $\mathbf{d}$ coincides with the positive z direction.

Let f be any facet of $\mathcal{P}$. We classify f, w.r.t. the given build direction $\mathbf{d}$, as a *front facet*, a *back facet*, or a *parallel facet* of $\mathcal{P}$, depending on whether the angle between the build direction $\mathbf{d}$ and the outward unit-normal, $\mathbf{n}_f$, of f is less than, greater than, or equal to 90°, respectively.

We now formalize the notion of supports. A facet of a polyhedron will need to be supported, if and only if, the angle between its outer normal and the build direction of the polyhedron is greater than 90°. This implies that the back facets of $\mathcal{P}$ will need to be supported. For concreteness, consider a back facet f of $\mathcal{P}$. The *support polyhedron* for f is the closure of the set of all points $p \in \mathbb{R}^3$ such that p is not in the interior of $\mathcal{P}$ and the ray shot from p in direction $\mathbf{d}$ first enters $\mathcal{P}$ through f. Informally, the support polyhedron of f is bounded from above by f, on the sides by vertical facets that "drop down" from the edges of f, and from below by the platform on which $\mathcal{P}$ rests and/or portions of front facets of $\mathcal{P}$. (If $\mathcal{P}$ is convex, then it is bounded from below by only the platform.) The *supports* of $\mathcal{P}$ w.r.t. a build direction is the collection of support polyhedra for the back facets.

The *support contact area* for $\mathcal{P}$ is the total surface area of $\mathcal{P}$ that is in contact with supports. It consists of the area of all the back facets of $\mathcal{P}$ and the areas of those portions of front facets and parallel facets that are in contact with supports. Note that for a convex polyhedron, the support structures are relatively

simple, in that only back facets are in contact with supports and every point on a back facet is in contact with supports. Furthermore, the support structures extend all the way down to the platform. However, for a general polyhedron, the situation is more complex: First, in addition to back facets, some front and parallel facets can also be in contact with supports (involuntarily, because of a back facet on a higher layer needing support). Second, front and parallel facets may be only partially in contact with supports. Finally, supports need not extend all the way down to the platform, but may, instead, terminate on other parts of the polyhedron itself. This is illustrated in Figure 1 for the supports for the fifth layer. (The figure is shown in 2D, for simplicity.) It is this complexity of the support structures that makes the support optimization problem that we consider so challenging.

3. AN UPPER BOUND ON THE QUALITY OF THE ESTIMATE

Ideally, we would like to find a direction $\mathbf{d}^*$ that minimizes the contact area of $\mathcal{P}$. Unfortunately, the structure of nonconvex polyhedra presents significant challenges in efficiently finding an optimal solution. Unlike convex objects, for which only the back facets are in contact with supports, nonconvex objects can have portions of front and parallel facets in contact with supports, as well. Furthermore, a small change in the build orientation can result in a significantly different footprint of the support structures, possibly affecting other facets that previously were not in contact with supports. These factors make it difficult to design an exact algorithm that is both efficient and practical. Therefore, a simple and efficient heuristic to estimate the contact area, along with a measure of the quality of the estimate, can be quite useful in practice.

We now develop a measure for the quality of the estimate provided by a given heuristic. Specifically, the ratio test developed below gives an indication of how close the estimated contact area is to the optimal contact area. Let $CA(\mathbf{d})$ denote the contact area of $\mathcal{P}$ for a given build direction, $\mathbf{d}$, and let $\hat{\mathbf{d}}$ be the direction computed by a heuristic. We show how to obtain an upper bound on the ratio $CA(\hat{\mathbf{d}})/CA(\mathbf{d}^*)$ via two relatively easy-to-compute quantities. Let $BFA(\mathbf{d})$ be the total area of the back facets w.r.t. $\mathbf{d}$ and let $\mathbf{d}'$ be a direction that minimizes the total area of back facets.

Notice that $BFA(\mathbf{d}^*) \leq CA(\mathbf{d}^*)$, since $CA(\mathbf{d}^*)$ includes possible contact area on front and parallel facets, and $BFA(\mathbf{d}') \leq BFA(\mathbf{d}^*)$, by definition of $\mathbf{d}'$. Therefore,

$$\frac{CA(\hat{\mathbf{d}})}{CA(\mathbf{d}^*)} \leq \frac{CA(\hat{\mathbf{d}})}{BFA(\mathbf{d}^*)} \leq \frac{CA(\hat{\mathbf{d}})}{BFA(\mathbf{d}')} \tag{1}$$

The above result allows us to upper-bound the contact area estimate for a set of candidate directions, relative to the (unknown) optimal solution, and to choose from these the best direction $\hat{\mathbf{d}}$. Notice that $BFA(\mathbf{d}')$ needs to be computed only once, and, therefore, the quality test will depend mainly on the efficiency of computing the contact area for a given direction. In the next section, we present two algorithms that compute the contact area, but differ in their accuracy and efficiency.

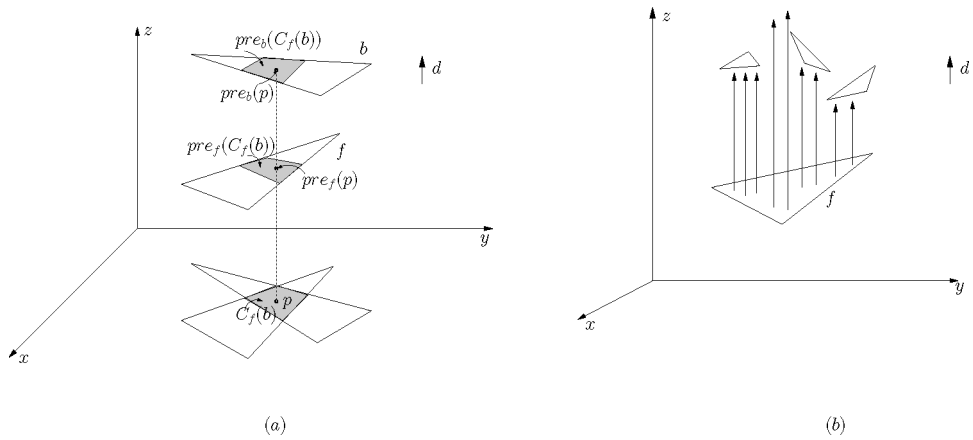


Fig. 2. (a) Computing a patch that is in contact with supports in the exact algorithm; (b) ray shooting for a front facet f in the heuristic.

We note that the upper bound in Eq. (1) is considerably weaker than a performance guarantee typically proved for an approximation algorithm. Proving such a bound would be desirable, but it has eluded us thus far. Nevertheless, the bound in Eq. (1) is still useful, since it can aid the user in selecting a suitable heuristic for estimating contact area, as we will see in Section 7.

4. COMPUTING CONTACT AREA ON FRONT FACETS FOR A FIXED DIRECTION

In this section, we describe an exact algorithm and a heuristic for computing the contact area on front facets for a fixed build direction **d**. The exact algorithm is simple, but too slow to be of practical use. However, this is tolerable, since we use the exact algorithm merely to do a one-time (offline) verification of the accuracy of our heuristic, which is simple and efficient.

4.1 An Exact Algorithm

W.l.o.g. assume that $\mathcal{P}$ rests on the xy plane and that the build direction **d** coincides with the positive **z** direction. Let f be a fixed front facet and let b be any back facet. We project f and b to the xy plane and compute the intersection of their projections (i.e., triangles), which yields a convex polygon, $C_f(b)$ (Figure 2(a)). If $C_f(b) \neq \emptyset$, then let p be any point in it, say the centroid. If the preimages, $pre_f(p)$ and $pre_b(p)$, of p on f and b, respectively, are such that $pre_b(p)$ is above $pre_f(p)$ in direction **d**, then $pre_f(p)$ is in contact with supports. This implies that the preimage $pre_f(C_f(b))$ is in contact with supports. This follows since no facet of $\mathcal{P}$ pierces another, so there cannot be another point q in $C_f(b)$, whose preimages on f and b are in the opposite order from those of p. (Note that it need not be the case that the cylinder bounded by $pre_f(C_f(b))$ and by $pre_b(C_f(b))$ is a support cylinder, since b, or parts thereof, need not be immediately above f; there could be parts of other back facets in between.) We can compute the portions of the front facet f that are in contact with supports, i.e.,

Table I. Comparison Between the Actual Running Times of the $O(n^3 \log n)$ Algorithm in Section 4.1 and the Algorithm in [Shaul and Halperin 2002][a]

Model (.stl)	Reduced #facets	Time (s.) Algo. in Section 4.1	Time (s.) Algo. in [Shaul and Halperin 2002]
mj	1896	3390	11649
triad1	1983	3788	15161
random tri. 1	400	3203	800
random tri. 2	500	6906	1129

[a]The first two test cases are decimated versions of real-world models while the last two are stacks of randomly generated overlapping triangles. All experiments were done on a sunBlade 100 machine with 512 MB of main memory and a 500-MHz processor; Exact arithmetic is used in both implementations.

the *footprint* of supports on f, by taking the union of the preimages $pre_f(C_f(b))$ that are found to be in contact with supports, for all back facets b. (In our implementation, we used the functions provided by LEDA [Mehlhorn and Naher 1999] to perform the union and intersection operations.)

The most expensive part of this algorithm turns out to be the union step in the computation of the footprints. Note that the algorithm simply projects all back facets down to the xy plane, without regard to any intervening facets. Thus, the complexity of the union of the polygons $pre_f(C_f(b))$ on a single front facet, f, can be $\Theta(n^2)$ in the worst case, and $\Theta(n^3)$ over all front facets. (An example of this is a configuration of $n/3$ front facets stacked on top of each other and two sets of $n/3$ back facets above them that overlap in the form of a trellis.) The total time to compute the union is $O(n^2 \log n)$ in the worst case for any front facet [de Berg et al. 1997], hence, $O(n^3 \log n)$ over all front facets. The storage requirement is $O(n^2)$; since the algorithm works on a facet-by-facet basis, the space can be reused.

With some additional effort, the running time can be improved to $O(n^3)$, as follows: For each front facet, f, we first project all back facets onto f. Next, we take the lines supporting each polygon in the projection, compute the arrangement of these lines, and determine the cells in this arrangement that are covered by the projection of at least one back facet. The union of these cells gives the footprint of the supports on f. The covering information for the cells can be computed incrementally, by doing a depth-first traversal of the dual graph of the arrangement and maintaining a counter that is incremented or decremented depending on whether or not the next cell in the traversal is covered by the triangle whose side is crossed to reach the cell. The time per front facet reduces to $O(n^2)$ and the claimed bound follows. (Note that this method, unlike the previous one, takes quadratic time per front facet, regardless of the geometric complexity of the footprint.)

We note also that a theoretically faster algorithm, running in $O(n^2 \log n)$ time and $O(n^2)$ space, is possible. This algorithm uses *cylindrical decomposition* [Mulmuley 1993] and respects intervening facets during projection. An output-sensitive algorithm with running time $O(n \log^2 n + V \log n)$, where V is the complexity of the decomposition, is given in [Shaul and Halperin 2002]. Unfortunately, the algorithms described in this section and in [Shaul and Halperin 2002] are extremely sensitive to degenerate input configurations,

and, therefore, reliable implementations require the use of exact arithmetic. Our experimental results indicate that the use of exact arithmetic introduces considerable overhead in the running time. Therefore, as mentioned at the beginning of Section 4, we use an exact-result algorithm merely for a one-time verification of the heuristic in Section 4.3. We have chosen to implement the relatively simpler $O(n^3 \log n)$-time algorithm described above for this purpose.

Table I shows the running time of our $O(n^3 \log n)$-time algorithm and that of [Shaul and Halperin 2002] on two groups of data. The first group consists of two geometric models.[1] The vertex coordinates of the two models have been slightly perturbed and parallel facets have been removed to satisfy the general position assumption required by the algorithm in [Shaul and Halperin 2002]; this explains why the number of facets in these models is slightly lower here than in Tables II and III (see later). The second group consists of sets of triangles whose coordinates have been generated randomly in a fixed range.

As can be seen from the table, our algorithm runs faster on the two models and slower on the sets of overlapping triangles than does the algorithm in [Shaul and Halperin 2002]. One possible explanation for this is that the triangle datasets contain many pairs that overlap in projection and represent "difficult" configurations for our algorithm. In real-world models, the number of overlapping triangles is generally much smaller and so our algorithm runs faster. (Our algorithm also does not make any general position assumptions.)

4.2 A Heuristic Based on Ray Shooting

In this section, we describe a simple heuristic with a very fast rate of convergence, which makes it practical for real data sets. Consider a front facet f and let p be a point on f. The point p will be in contact with supports if, and only if, the ray originating at p in direction $\mathbf{d}$ intersects some other facet of $\mathcal{P}$. Thus, the main idea behind the heuristic is to pick a set of points in the interior of f and identify those points that will be in contact with supports by shooting rays in direction $\mathbf{d}$ (Figure 2(b)). Let H_f (resp. M_f) denote the set of rays, originating at points on f, that hit a facet of $\mathcal{P}$ (resp. miss all facets of $\mathcal{P}$). Then, the area of f that is in contact with supports can be estimated as $(|H_f|/(|H_f| + |M_f|)) * area(f)$. As the density of the sample points is increased, the accuracy of the estimate improves (Figure 2(b)).

The sample points are selected through an adaptive subdivision process. During the execution of the algorithm, each front facet is subdivided into a number of triangular patches (initially the entire facet is the only patch). The centroids of the patches are selected as sample points and the results from the ray shooting are recorded. Next, each patch is subdivided into two triangles (for example, by connecting the midpoint of its longest side with the opposite vertex). The new patches are placed at the end of a queue of unprocessed patches, which guides the subdivision process in a breadth-first search fashion.

[1]The geometric models used here (and those used later in Tables II and III) are real-world models that have been reduced in size, while preserving their general topology, in order to keep the running times reasonable. We use software such as Decimator and VRMesh [VRMesh] to do the reduction. However, our final experiments in Tables IV and V use the original models.

The subdivision process is made adaptive in that the number of sample points per facet is weighted based on area, so that larger facets have more sample points. In a preprocessing step, each front facet is subdivided repeatedly until the resulting patches have areas that are less than the average of the areas of the front facets of $\mathcal{P}$. This ensures that larger facets are subdivided deeper and, consequently, generate more sample points for the ray-shooting phase. Furthermore, as the following lemma shows, the preprocessing does not result in too many patches.

LEMMA 4.1. *The number of patches generated after the preprocessing step is less than* $3n$*, where* n *is the number of facets of* $\mathcal{P}$.

PROOF. Let m be the number of front facets of $\mathcal{P}$, A be total area of the front facets, and $\bar{A} = \frac{A}{m}$ be the average front facet area. Let $\mathcal{S}$ be the set of patches that result from the front facets of $\mathcal{P}$ that are subdivided at least once. Since our method of subdivision halves the area of a patch and the area is larger than $\bar{A}$ just prior to the last subdivision, all patches in $\mathcal{S}$ have areas that are greater than $\bar{A}/2$. If we let A' denote the total area of the patches in $\mathcal{S}$, and let $k = |\mathcal{S}|$, then $k\frac{\bar{A}}{2} < A'$. Since $A' \leq A$, we have $k\frac{\bar{A}}{2} < A$, or $k\frac{A}{2m} < A$, and, therefore, $k < 2m$.

In addition, there are, at most, m front facets of $\mathcal{P}$ that are not subdivided in preprocessing (because their areas are less than $\bar{A}$ to begin with). Since $m \leq n$, the lemma follows. □

Each iteration of the algorithm corresponds to a complete subdivision of the patches and, therefore, will process twice as many patches as the previous one. The algorithm terminates after a predefined number of iterations, controlled by the user, or when the change in contact area is not significant. Currently, we use the convergence criterion $\delta = |CA_{i+1} - CA_i| < 0.01 * CA_i$, i.e., the algorithm terminates when the contact area from iteration i to iteration $(i + 1)$ (denoted by CA_i and CA_{i+1}, respectively) changes by less than 1%.

During each iteration i, $i \geq 1$, the algorithm processes less than $2^{i-1} * 3n$ patches. In our implementation, a brute-force approach is used to answer a ray-shooting query in $O(n)$ time. Thus, for each patch, the ray shooting takes $O(n)$ time to decide whether the ray hits any facet of $\mathcal{P}$, so the overall time per iteration is $O(2^i * n^2)$. Therefore, for a user-specified number, d, of iterations, the overall running time is $O(2^d * n^2)$.

4.3 Experimental Results

Table II provides a comparison between the exact algorithm and the heuristic. The heuristic was run with two different terminating criteria: (1) terminating after ten iterations, denoted as "$\tau = 10$", following the initial subdivision of large facets; (2) terminating based on the convergence criterion described in Section 4.2, denoted as "$\delta < 1\%$." In the latter case, we also imposed an upper limit of ten iterations following the initial subdivision of large facets, so that the computation did not become prohibitively expensive. The comparison tests were run on decimated versions of real-world STL models because of the slow performance of the exact algorithm from Section 4.1 that we implemented for

Table II. Comparison Between the Exact Algorithm and the Heuristic to Compute, for a Given Direction, the Contact Area on Front Facets[a]

Model (.stl) (#facets)	Reduced #facets	Algorithm	Contact Area (% difference)	Time (s.)
bot_case (17642)	2000	$\tau = 10$	13644.0 (0%)	41
		$\delta < 1\%$	13660.6 (0%)	1
		exact	13642.3	4182
carcasse (22876)	2000	$\tau = 10$	63.75 (0%)	34
		$\delta < 1\%$	63.61 (0%)	1
		exact	63.73	3296
mj (2832)	2000	$\tau = 10$	1.68 (0%)	29
		$\delta < 1\%$	1.66 (−1%)	1
		exact	1.68	3391
top_case (16692)	2000	$\tau = 10$	10239.5 (0%)	53
		$\delta < 1\%$	10251.5 (0%)	1
		exact	10268.1	4790
triad1 (11352)	2000	$\tau = 10$	0.33 (0%)	36
		$\delta < 1\%$	0.33 (0%)	1
		exact	0.33	3757

[a]Large models have been decimated to 2000 facets because of the slow speed of the exact algorithm. Note that the exact algorithm is only used for a one-time verification of the accuracy of the heuristic. The models bot_case and top_case required two iterations after the initial subdivision to meet the $\delta < 1\%$ terminating criterion; carcasse and mj required three iterations; triad1 required four iterations. In the column labeled "contact area", the number in parenthesis shows the percentage difference between the contact area computed by the heuristic and by the exact algorithm (values are rounded to the nearest percent). The z direction is chosen as a build direction. The models used here and in the rest of the tables are illustrated (later) in Table V.

purposes of comparison. (However, our final experiments in Tables IV and V were done on original models, not decimated ones.) As can be seen, the heuristic provides nearly the same answer as the exact algorithm, but in a fraction of the time. All experiments were done on a SunBlade 100 machine with 512 MB of main memory and a 500-MHz processor. Programs were written in C++ and use CGAL and LEDA [Mehlhorn and Naher 1999]; the code for the heuristic uses floating-point arithmetic.

5. HANDLING PARALLEL FACETS

The previous algorithms consider only the contact area on front and back facets. In order to get an overall estimate of the contact area, we need to also consider the portions of parallel facets that are in contact with supports. In Section 5.1, we give an algorithm for computing the exact contact area on parallel facets; in Section 5.2, we give an efficient heuristic, analogous to the one in Section 4.2.

5.1 Exact Algorithm

Let f be a parallel facet and let V_f be the vertical strip that is in the supporting plane of f and contains f exactly. We may assume, w.l.o.g., that no vertex of f is in the interior of V_f. (Each bounding line of V_f contains at least one vertex of f. If there is a vertex in the interior of V_f, we draw a vertical line through

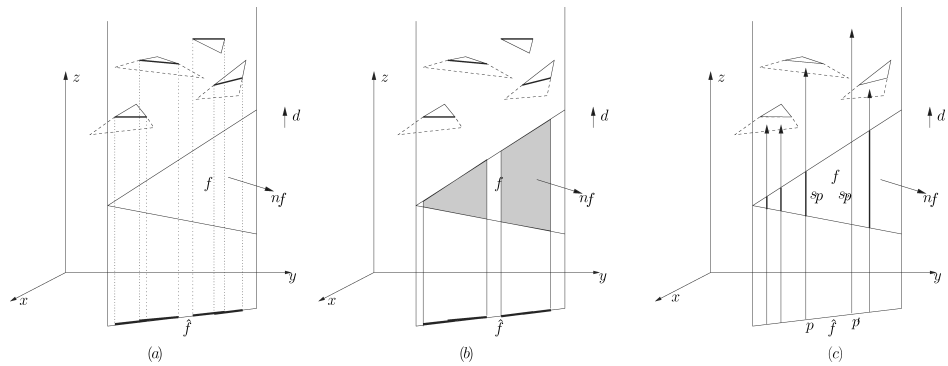


Fig. 3. Computing contact area on parallel facets: (a) (exact algorithm) identifying the segments that require supports; (b) (exact algorithm) identifying the portions of f that are in contact with supports; (c) (heuristic) segment s_p is in contact with supports, but $s_{p'}$ is not.

it and split f into two facets that each satisfy the assumption.) Let $\hat{f}$ be the projection of f on the xy plane. (Notice that $\hat{f}$ is just a line segment.)

Consider the back facets of $\mathcal{P}$ that either pierce V_f above f, or touch V_f above f and are in the same halfspace of V_f as the outer unit normal, $\mathbf{n}_f$, of f. (These are the back facets whose supports are potentially in contact with f when $\mathcal{P}$ is built in direction $\mathbf{d}$.) The intersections of these back facets with V_f is a set, A, of line segments (Figure 3(a)).

Let A' be the set of segments that correspond to the projections of the segments in A on the xy plane. Clearly, all the segments in A' lie on $\hat{f}$ and can be merged efficiently, so that no two of the resulting segments overlap. For each merged segment s, we erect a vertical strip V_s and find its overlap area with f (Figure 3(b)). The sum of the areas of overlap for all strips V_s gives the overall contact area on f.

For each facet f, the size of A is $O(n)$. Merging the segments in A' can be done efficiently in $O(n \log n)$ time by presorting them on their first endpoint. For each strip V_s the overlap area can be found in constant time. Therefore, the algorithm takes $O(n \log n)$ time per parallel facet, or $O(n^2 \log n)$ time for all parallel facets.

5.2 Heuristic

Let f be a parallel facet and let $\hat{f}$ be the projection of f on the xy plane. (Notice that $\hat{f}$ is just a line segment.) Let p be any point on $\hat{f}$ and let s_p be the segment obtained by intersecting f with the ray originating at p in direction $\mathbf{d}$. The segment s_p will be in contact with supports if, and only if, the supporting line of s_p intersects properly a facet of $\mathcal{P}$ at a point above s_p (Figure 3(c)).

Let S be a set of sample points on $\hat{f}$. Let H_f (resp. M_f) be the set of segments, s_p, that are (resp. are not) in contact with supports. If $length(H_f)$ and $length(M_f)$ denote the sum of the lengths of the segments in H_f and M_f, respectively, we can estimate the contact area on f as $(length(H_f)/(length(H_f) + length(M_f))) * area(f)$. As the number of sample points is increased, the accuracy of the estimate also increases.

Table III. Comparison between the Exact Algorithm and the Heuristic to Compute, for a Given Direction, the Contact Area on Parallel Facets of the Models in Table II[a]

Model (.stl) (#facets)	Reduced #facets	Algorithm	Contact Area (% difference)	Time (s.)
bot_case (17642)	2000	$\tau = 10$	701.17 (0%)	0
		$\delta < 1\%$	701.17 (0%)	0
		exact	701.17	1
carcasse (22876)	2000	$\tau = 10$	22.70 (0%)	8
		$\delta < 1\%$	22.70 (0%)	0
		exact	22.68	143
mj (2832)	2000	$\tau = 10$	1.88 (0%)	4
		$\delta < 1\%$	1.88 (0%)	0
		exact	1.88	40
top_case (16692)	2000	$\tau = 10$	4352.26 (0%)	21
		$\delta < 1\%$	4340.39 (0%)	0
		exact	4352.30	352
triad1 (11352)	2000	$\tau = 10$	0.065 (0%)	3
		$\delta < 1\%$	0.065 (0%)	0
		exact	0.065	34

[a]The z direction is chosen as a build direction. Four of the models required two iterations after the initial subdivision to meet the $\delta < 1\%$ terminating criterion; carcasse required three iterations. In the column labeled "contact area" the number in parenthesis shows the percentage difference between the contact area computed by the heuristic and by the exact algorithm (values are rounded to the nearest percentage).

The sample points are selected through an adaptive subdivision process similar to the one described in Section 4.2. The main difference is that the patches are line segments, and not triangles (initially, each facet f is represented by the patch $\hat{f}$). The sample points are the midpoints of the corresponding patches and each patch is subdivided into two equal length segments at its midpoint. In our implementation a patch is represented by the edge of f that completely spans the strip V_f, i.e., whose projection on the xy plane is the same as $\hat{f}$. The patches are subdivided until the lengths of the span segments fall below the initial average length of all span segments. The terminating criterion is the same as the one described in Section 4.2. Table III summarizes the execution of the exact algorithm and the heuristic on decimated models.

6. MINIMIZING BACK FACET AREA

The efficiency of computing the upper-bound in Eq. (1) depends critically on the efficiency of finding a direction $\mathbf{d}'$ that minimizes the back facet area. In this section we describe an algorithm, based on arrangements of great circles on the unit sphere, that computes $\mathbf{d}'$.

6.1 Preliminaries

Let $\mathcal{S}^2$ denote the unit sphere of directions. We map each facet, f, to a point on $\mathcal{S}^2$ corresponding to the unit vector, $\mathbf{n}_f$, normal to the supporting plane of f. Let $\mathcal{C}_f$ be the set of points on $\mathcal{S}^2$ that are at distance $\pi/2$ from $\mathbf{n}_f$, i.e., $\mathcal{C}_f$ is a great circle on $\mathcal{S}^2$. (Note that several facets of $\mathcal{P}$ can correspond to a single great

circle.) $\mathcal{C}_f$ defines two open hemispheres: $\mathcal{H}_f^+$ with pole $\mathbf{n}_f$, and $\mathcal{H}_f^-$ with pole $-\mathbf{n}_f$. Given a build direction $\mathbf{d}$ the facet f will be a *back* facet and, therefore, will require supports, if, and only if, $\mathbf{d} \in \mathcal{H}_f^-$. Similarly, f will be a *front* facet, requiring no supports, if, and only if, $\mathbf{d} \in \mathcal{H}_f^+$. Finally, f will be a *parallel* facet, requiring no supports, if, and only if, $\mathbf{d} \in \mathcal{C}_f$. (Although a front or parallel facet requires no support, it could be in contact with supports required by back facets, as seen previously.)

Consider the arrangement, $\mathcal{A}$, of great circles $\mathcal{C}_f$ corresponding to the facets f of $\mathcal{P}$. $\mathcal{A}$ decomposes $\mathcal{S}^2$ into three types of *elements*: (1) *cells*, which are (open) regions of intersection of the hemispheres defined by the great circles, (2) *arc edges*, which are (open) portions of great circles and determine the boundaries of the cells in $\mathcal{A}$, and (3) *vertices*, which are intersections of great circles and are the endpoints of arc edges.

LEMMA 6.1. *The elements of $\mathcal{A}$ define regions on $\mathcal{S}^2$ that correspond to sets of directions for which the back facet area is constant.*

LEMMA 6.2. *The build direction $\mathbf{d}'$ minimizing the back facet area corresponds to a vertex in $\mathcal{A}$.*

PROOF. We show that the back facet area corresponding to any point in a cell or arc edge is never less than the back facet area corresponding to the vertices of the cell. This implies that it is sufficient to consider only the vertices of $\mathcal{A}$ in order to find $\mathbf{d}'$.

Let c be a cell in $\mathcal{A}$. For any element q of $\mathcal{A}$, let $BFA(q)$ be the back facet area associated with any direction in q. By Lemma 6.1, $BFA(q)$ is well-defined and a constant. Notice that the edges bounding c represent transitions onto great circles, which correspond to front and/or back facets becoming parallel facets. Therefore, the set of back facets corresponding to any point on the boundary of c is either the same as or a proper subset of the set of back facets corresponding to any point in the interior of c. This implies that the back facet area cannot increase. Therefore, $BFA(e) \leq BFA(c)$ for any edge e on the boundary of c.

Let e be an edge in $\mathcal{A}$ and let u be one of the vertices in $\mathcal{A}$ that is adjacent to e along the supporting great circle $\mathcal{C}(e)$ of e. The vertex u represents a transition, along the arc edge e, onto a great circle other than $\mathcal{C}(e)$ along the arc edge e. Thus, a front and/or a back facet becomes parallel. Arguing as before, $BFA(u) \leq BFA(e)$.

The above discussion shows that the back facet area corresponding to any point within a cell in $\mathcal{A}$ is never less than the back facet area corresponding to any point along the bounding edges of the cell. Furthermore, the latter is never less than the back facet area at the vertices in $\mathcal{A}$ adjacent to the edge. Thus, in order to identify the direction $\mathbf{d}'$ that minimizes the back facet area, it is sufficient to examine only the directions corresponding to the vertices in $\mathcal{A}$. □

6.2 The Algorithm

Lemma 6.2 shows that to find the direction, $\mathbf{d}'$, that minimizes the back facet area, it is sufficient to consider only the directions on $\mathcal{S}^2$ that correspond to the

vertices of $\mathcal{A}$. This immediately suggests an algorithm for finding $\mathbf{d}'$:

- (*preprocessing*) Compute the arrangement $\mathcal{A}$ of great circles on the unit sphere.
- (*initialization*) Let u be any vertex in $\mathcal{A}$. Identify the front, back, and parallel facets determined by the direction corresponding to u and initialize the back facet area term to the total area of the back facets.
- (*update*) Walk along the vertices of the arrangement, by visiting adjacent vertices connected by an arc edge. Notice that a vertex in $\mathcal{A}$ is the intersection of great circles and each great circle describes a set of directions for which front and/or back facets become parallel. Therefore, during the transition from vertex u to vertex v, let $\Delta BFA(u)$ be the area of the parallel facets at u that become back facets at v and let $\Delta BFA(v)$ be the area of the parallel facets at v that were back facets at u. Then, $BFA(v) = BFA(u) + \Delta BFA(u) - \Delta BFA(v)$.
- During the walk along the vertices of the arrangement, we keep track of the vertex v for which the back facet area is minimized and report as $\mathbf{d}'$ the direction corresponding to v.

The *preprocessing* step of the above algorithm takes $O(n^2)$ time and $O(n^2)$ space, since the number of great circles in the arrangement is $O(n)$. At each vertex in the arrangement, we spend time proportional to the degree of the vertex and, therefore, the overall time during the *update* step of the algorithm is $O(n^2)$. Therefore, the algorithm takes $O(n^2)$ time and uses $O(n^2)$ space.

The space usage of the algorithm can be improved to $O(n)$ at the expense of increased running time to $O(n^2 \log n)$. The main idea is to walk along arc edges belonging to the same great circle. This allows us to focus on only a portion of the arrangement $\mathcal{A}$. Given a great circle $\mathcal{C}_f$, we compute its intersections with all the other great circles and sort the vertices of intersection in their circular order along $\mathcal{C}_f$. Next, we pick an arbitrary vertex and initialize the back facet area term. Finally, we visit all the vertices along $\mathcal{C}_f$ and update the back facet area term following the rule described in the *update* step of the algorithm above. During the walk, we keep track of the vertex, v, corresponding to the direction for which the back facet area is minimized. The optimal direction is identified after all great circles have been processed.

The running time per great circle is dominated by the time to sort $O(n)$ vertices of intersection in time $O(n \log n)$. The walk along a great circle spends $O(n)$ time for the initialization and constant time per vertex, or $O(n)$ in total. Over all great circles, the running time is $O(n^2 \log n)$. Since we do not compute the whole arrangement, the space is $O(n)$ and this can be reused.

Table IV summarizes the results of the execution of the $O(n^2 \log n)$-time algorithm on nondecimated models.

We remark that it is possible to obtain a slightly more efficient algorithm ($O(n^2)$ time and $O(n)$ space) at the expense of increased algorithmic complexity. For convenience, we map the portions of the great circles lying in the upper half of $\mathcal{S}^2$ to straight lines using *central projection* [Preparata and Shamos 1993]. We can compute the back facet areas at the vertices of the resulting planar arrangement, which are in 1-1 correspondence with the vertices of the

Table IV. Performance of the Algorithm for Computing a Direction that Minimizes the Area of the Back Facets

Model (.stl)	#Facets	Minimum area (back facets)	Time (s.)
bot_case	17642	14409.4	4079
carcasse	22876	35.98	6784
mj	2832	5.81	82
top_case	16692	7843.5	3446
triad1	11352	2.20	1581

arrangement in the upper half of $\mathcal{S}^2$, by doing a sweep: when the sweep reaches a vertex v, we compute $BFA(v)$ from $BFA(u)$, as above, where u is a neighbor of v that has already been visited. However, to obtain the desired $O(n)$ space bound, we cannot afford to do a traditional sweep, which requires the entire arrangement to be precomputed and stored. Instead, we use the *topological sweep* method [Edelsbrunner and Guibas 1989] which computes and retains only the portion of the arrangement that is relevant currently—this has size $O(n)$. A similar approach is used for the lower half of $\mathcal{S}^2$. We note that this approach was used previously in [Majhi et al. 1999a] to minimize contact area of supports for a convex polyhedron.

7. APPROXIMATING THE CONTACT AREA

In this section, we present several heuristics for choosing a candidate build direction to estimate the optimal contact area requirements. The quality of each heuristic is measured in terms of the ratio $CA(\hat{\mathbf{d}})/BFA(\mathbf{d}')$, where $\hat{\mathbf{d}}$ is the direction computed by the heuristic, and $\mathbf{d}'$ is the direction that minimizes the area of the back facets. As shown in Section 3 (Eq. 1) this ratio is an upper bound on $CA(\hat{\mathbf{d}})/CA(\mathbf{d}^*)$, where $\mathbf{d}^*$ is the direction that minimizes the overall contact area.

We have implemented and tested the following choices for build direction:

- *min BFA direction*—direction that minimizes the back facet area, as discussed in Section 6. Since the overall contact area includes the area of the back facets, it may be advantageous to choose a direction that results in low contact area contribution from the back facets.
- *max PFA direction*—direction that maximizes the area of parallel facets. This direction is computed along with the direction that minimizes the back facet area; it is easy to adapt the proof of Lemma 6.2 to show that the same candidate directions need to be examined in both cases. We consider the direction and its opposite, since both generate the same area of parallel facets, and take the better result. The intuition behind this heuristic is that parallel facets do not themselves require supports and, therefore, by maximizing the area of parallel facets the number of support structures could be reduced, which could lead to reduced amount of contact area.
- *max PFC direction*—direction that maximizes the count of parallel facets. This direction is computed along with the direction that minimizes the back

facet area; again, it is easy to adapt the proof of Lemma 6.2 to show that the same candidate directions need to be examined in both cases. We consider the direction and its opposite, since both generate the same count of parallel facets, and take the better result. This is an alternative to the previous heuristic, but we try to maximize the number of facets that will not require direct supports, which could lead to a reduction in support structures and, therefore, a reduction in the amount of contact area.

- *PC direction*—direction that corresponds to the principal components of the object. Intuitively, this heuristic builds the object along one of three mutually perpendicular axes that capture the relative shape of the object. We consider each direction and its opposite and take the best result. The principal component directions were computed using MATLAB software from The MathWorks, Inc.
- *Flat direction*—direction that is opposite to the outward unit normal of a facet of the model. During the build phase it is often desirable to build the part such that it rests on one of its facets. In this case, the facet must be contained in the boundary of the convex hull, of the model (notice that a facet on the convex hull may contain several facets from the original model). We select the facet, f, on the convex hull, which contains facets from the original model that have the largest total area and use $-\mathbf{n}_f$ as the build direction, where $\mathbf{n}_f$ is the outward unit normal of f. The convex hull is computed using the functionality provided by the CGAL library. (Note that this heuristic is somewhat similar to the one in Allen and Dutta [1995], which is described in Section 1. Unfortunately, we have not been able to make a direct comparison of the two heuristics as the software for the one used in Allen and Dutta [1995] appears to be no longer available [personal communication from D. Dutta].)
- *Random directions*—directions chosen at random. This heuristic was included for comparison purposes only. We chose a set of fifteen random directions, computed the contact areas for each of these directions, took their mean, and then divided this by the minimum back facet area to arrive at the mean contact area ratio.

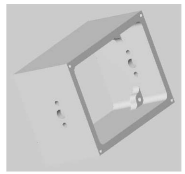
Table V illustrates the models used for our experiment and summarizes the results. For each model, the table shows the contact area ratio computed for each heuristic and compares the ratio realized by the best heuristic with the ratio realized by the random heuristic. As seen in the column named "comparison," savings ranging from 9 to 83% are achieved on real-world (nondecimated) models. Note that even though we are comparing ratios, this is equivalent to comparing the contact areas themselves, since the denominators for both ratios are the same, namely, the minimum back facet area.

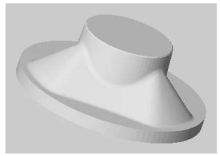
How might a designer use the results in Table IV? Suppose that the designer wishes to fabricate model `oldbasex`, using no more than twice the minimum contact area. Table V shows that only "Max PFA ratio" guarantees that this requirement will be met, so the designer can proceed to safely use this heuristic. (It is possible that some of the other heuristics will also meet the requirement, since the upper-bound in Eq. 1 is loose, but this is not guaranteed.)

Table V. Performance of the Heuristics for Estimating Contact Area[a]

prism1

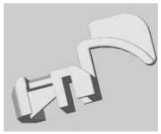
pyramid1

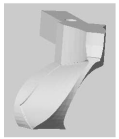

stlbin2


ecc4

triad1

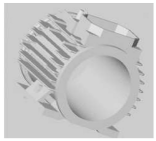
tod21

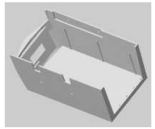

f0m27

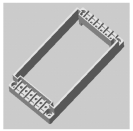

mj

3857438

top_case


carcasse


cover-5


oldbasex

bot_case

Model (.stl)	#Facets	Max Possible Ratio	Min BFA Ratio	Max PFA Ratio	Max PFC Ratio	PC Ratio	Flat Ratio	Random Dir. (mean) Ratio	Comparison (%)	Time (s.)
prism1	20	55.2	1.0[1]	1.0[1]	24.0[4]	1.0[1]	24.0[4]	30.3[6]	97	4
pyramid1	10	14.3	1.0[1]	3.7[3]	6.0[4]	1.0[1]	6.0[4]	7.8[6]	87	2
stlbin2	2761	15.8	2.2[1]	2.2[1]	9.2[4]	8.5[3]	9.2[4]	9.8[6]	77	90
ecc4	4994	4.9	1.2[1]	1.2[1]	1.4[1]	1.9[5]	1.8[4]	2.7[6]	56	837
triad1	11352	2.9	1.9[4]	2.1[5]	2.1[5]	1.4[1]	1.4[1]	1.8[3]	19	2280
tod21	1128	7.2	1.1[1]	1.1[1]	3.8[4]	3.8[4]	1.1[1]	4.2[6]	75	49
f0m27	3730	4.3	2.4[2]	2.3[1]	2.3[1]	2.4[2]	3.2[6]	2.8[5]	17	213
mj	2832	5.3	2.1[1]	2.4[2]	2.4[2]	2.6[5]	2.4[2]	2.9[6]	27	124
3857438	12184	3.4	2.6[6]	2.5[3]	2.5[3]	2.4[2]	2.3[1]	2.5[3]	9	2576
top_case	16692	4.1	2.8[4]	2.8[4]	3.0[6]	2.1[2]	2.0[1]	2.5[3]	22	4944
carcasse	22876	6.2	3.5[3]	3.3[1]	4.0[4]	4.4[5]	3.3[1]	4.9[6]	34	9434
cover-5	906	5.7	3.9[3]	3.9[3]	3.9[3]	3.1[1]	5.2[6]	3.8[2]	19	10
oldbasex	3660	15.6	3.3[2]	1.7[1]	12.0[5]	8.2[3]	12.0[5]	10.4[4]	83	161
bot_case	17642	3.0	2.1[4]	2.1[4]	2.1[4]	1.5[2]	1.3[1]	1.9[3]	33	5617

[a]The first two models were hand-generated; the remaining models are from Stratasys, Inc, a leading manufacturer of LM machines (www.stratasys.com). All models are originals, not decimated ones. The numbers in square brackets show the rankings of the heuristics for each model. The column "comparison" shows the percentage improvement of the ratio achieved by the top-ranked heuristic over the mean ratio given by the random-directions heuristic. The column "max possible ratio" is given only as a reference and represents the extreme situation where all of the surface area of the object is in contact with supports. The column "time" includes the time for computing the minimum back facet area and the times for computing the ratios for all heuristics, except for the random heuristic, which is for purposes of comparison only. The time-consuming step is the computation of the back facet area.

8. CONCLUSION

We have presented a set of efficient and practical heuristics for estimating the contact area of supports for polyhedral models in layered manufacturing. We have also shown how the quality of the estimate, w.r.t. the unknown and hard-to-compute optimal solution, can be upper-bounded as the ratio of two relatively easy-to-compute quantities. Our algorithms have been implemented and tested on a range of real-world models and have been shown to perform well in practice.

An interesting problem for further work is computing a build direction that estimates the minimum volume of the support structures. Our approach in Section 3 for upper-bounding the contact area estimate does not appear to extend to the volume version of the problem, so a different approach may be needed.

ACKNOWLEDGMENTS

We thank the four reviewers for many useful suggestions. We are also grateful to the editor, John Hershberger, for an exceptionally thorough reading of the paper and for many thoughtful comments. All of this feedback has resulted in a much-improved paper.

REFERENCES

AGARWAL, P. AND DESIKAN, P. 2000. Approximation algorithms for layered manufacturing. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*. 528–537.

ALLEN, S. AND DUTTA, D. 1995. Determination and evaluation of support structures in layered manufacturing. *Journal of Design and Manufacturing 5*, 153–162.

ASBERG, B., BLANCO, G., BOSE, P., GARCIA-LOPEZ, J., OVERMARS, M., TOUSSAINT, G., WILFONG, G., AND ZHU, B. 1997. Feasibility of design in stereolithography. *Algorithmica 19*, 61–83.

BAUMGART, B. 1975. A polyhedron representation for computer vision. In *Proceedings of AFIPS National Computer Conference*. Vol. 44. AFIPS Press, Alrington, VA, 589–596.

BOSE, P. 1995. Geometric and computational aspects of manufacturing processes. Ph.D. thesis, School of Computer Science, McGill University, Montréal, Canada.

Computational Geometry Algorithms Library (CGAL). http://www.cgal.org.

CHUA, C., LEONG, K., AND LIM, C. 2003. *Rapid Prototyping: Principles and Applications*. World Scientific Publ. Inc., Singapore.

DE BERG, M., VAN KREVELD, M., OVERMARS, M., AND SCHWARZKOPF, O. 1997. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin.

Decimator™ 1.0, Raindrop Geomagic, Inc., North Carolina. http://www.geomagic.com/products/decimate/.

EDELSBRUNNER, H. AND GUIBAS, L. 1989. Topologically sweeping an arrangement. *Journal of Computer and System Sciences 38*, 165–194.

JACOBS, P. 1992. *Rapid Prototyping & Manufacturing: Fundamentals of StereoLithography*. McGraw-Hill, New York.

JOHNSON, E. 1999. Support generation for three-dimensional layered manufacturing. Master's project report, Dept. of CS & E, Univ. of Minnesota, Minneapolis, MN.

MAJHI, J. 1998. Geometric Methods in Computer-Aided Design and Manufacturing. Ph.D. thesis, Dept. of Computer Science & Engineering University of Minnesota, Minneapolis, MN.

MAJHI, J., JANARDAN, R., SCHWERDT, J., SMID, M., AND GUPTA, P. 1999a. Minimizing support structures and trapped area in two-dimensional layered manufacturing. *Computational Geometry: Theory & Applications 12*, 241–267.

MAJHI, J., JANARDAN, R., SMID, M., AND GUPTA, P. 1999b. On some geometric optimization problems in layered manufacturing. *Computational Geometry: Theory & Applications 12*, 219–239.

MEHLHORN, K. AND NAHER, S. 1999. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge.

MULMULEY, K. 1993. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice Hall, Englewood Cliffs, NJ.

MCMAINS, S. AND SÉQUIN, C. 1999. A coherent sweep plane slicer for layered manufacturing. In *Proceedings of the 5th ACM Symposium on Solid Modeling and Applications*. 285–295.

PREPARATA, F. AND SHAMOS, M. 1993. *Computational Geometry: An Introduction*. Springer-Verlag, New York.

SCHWERDT, J. 2001. Ph.D. thesis. Ph.D. thesis, Department of Computer Science, University of Magdeburg, Magdeburg, Germany.

SHAUL, H. AND HALPERIN, D. 2002. Improved construction of vertical decompositions of 3D arrangements. In *Proc. 18th ACM Symposium on Computational Geometry*. 283–292.

VRMesh™, VirtualGrid, Inc., Washington. http://www.vrmesh.com.

Received December 2003; accepted August 2006