Approximation Algorithms for Layered Manufacturing*

Pankaj K. Agarwal† Pavan K. Desikan‡

Abstract

An important problem in *layered manufacturing* is the choice of a good build direction, which influences the quality and the cost of manufacturing the object. We present efficient algorithms for computing a build direction that optimizes the total area of faces that are in contact with supports. For a given convex polytope with n faces and for a parameter $\varepsilon > 0$, we present an $O((n/\varepsilon^3)\operatorname{polylog} n)$ algorithm that computes a build direction so that the total area of faces in contact with supports is at most $(1+\varepsilon)A^*$, where A^* is the minimum area of contact with supports for all build directions. For non-convex polyhedra, we provide evidence that the lower bound for any algorithm computing the optimal direction might be $\Omega(n^4)$. We also present a faster algorithm for some special cases.

Our technique for convex polyhedra involves computing approximate levels in arrangement of lines with weights. For given parameters $\omega, \varepsilon > 0$, we present an algorithm that computes a $(1+\varepsilon)$-approximate weighted ω-level in an arrangement of n weighted lines in time $O((n/\varepsilon^3)\operatorname{polylog} n)$.

1 Introduction

Layered Manufacturing is the basis for an emerging technology called *rapid prototyping and manufacturing* (RP&M). This technology automatically generates physical objects layer-by-layer directly from the 3D-CAD data, without conventional manufacturing and tooling processes. This ability to create rapidly a physical version of the model helps in detecting and correcting flaws in the model in the early stages of the design cycle. This technology, which is used in automotive, aerospace and medical industries, accelerates dramatically the time it takes to bring a product to the market [19, 20].

In this paper we consider one of the methods for layered manufacturing called *stereolithography*. In this method for RP&M, the model is generated layer by layer, using a laser [19]. Ideally each new layer should rest in its entirety on the previous layer [20]. In real models, however, portions of a slice can overhang the previous slice. Hence additional structures called *supports* are generated and merged into the model; see Figure 1. In the postprocessing steps, these supports are removed. The amount of postprocessing time and the quality of finish depend on the area of the model surface that is in contact with support. Hence, choosing a build direction that minimizes this contact area of support is an important optimization problem in layered manufacturing. The other optimization criteria that can be considered include minimizing the volume of support, or the *stair-step error* which is the error in faces that are not parallel to the build direction because of the minimum layer thickness.

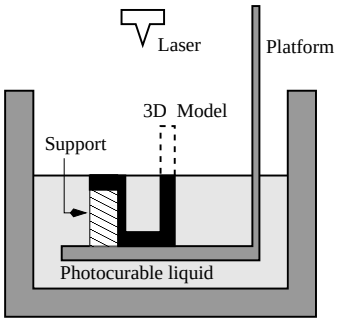

Figure 1. The stereolithography apparatus.

Previous results: Motivated by these applications, there has been recent work on geometric algorithms in layered manufacturing [5, 19, 21, 26]. If we want to build a convex polyhedron $\mathcal{P}$, it can be shown that the contact area of support is the sum of the areas of all the downward facing faces of $\mathcal{P}$. For non-convex polyhedra, some upward facing facets may also requiring supports. For convex polyhedra either the entire face requires support or the face does not require support at all. For non-convex polyhedra it is possible that only a part of the face is in contact with supports. For convex polyhedra, Majhi *et al.* [20] gave an $O(n^2)$ algorithm for computing a direction that minimizes the contact area of support. When the model is built on one of the faces of the polyhedron $\mathcal{P}$, they give a close to $O(n^{4/3})$-time algorithm for computing a build direction that minimizes the area of support. The problem of minimizing the stair-step error was also

*Work on this paper was supported by Army Research Office MURI grant DAAH04-96-1-0013, by a Sloan fellowship, by NSF grants EIA–9870724 and CCR–9732787 and by a grant from the U.S.-Israeli Binational Science Foundation.

†Center for Geometric Computing, Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129, USA. E-mail: pankaj@cs.duke.edu

‡Center for Geometric Computing, Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129, USA. E-mail: pkd@cs.duke.edu

considered by Majhi *et al.* [20], who reduced the problem to that of computing the convex hull a point set in $\mathbb{R}^3$, and obtained an $O(n \log n)$ algorithm for computing the direction that minimizes the maximum stair-step error. They also gave an $O(n^2)$-time algorithm for computing a direction that minimizes the sum of stair-step errors. A direction that minimizes the volume of supports for convex polyhedra can also be found in time $O(n^2)$ [20]. Some algorithms for optimizing two or more of these optimization criteria simultaneously were also presented in [21].

Our results: All known solutions for computing the best build direction are quite expensive. In this paper, we develop approximation algorithms for computing a build direction that minimizes the area of the faces that are in contact with supports. For a convex polyhedron, given $\varepsilon > 0$, we describe a randomized algorithm for computing a build direction $\mathbf{n}$, for which the contact area of support in the direction $\mathbf{n}$ is $(1+\varepsilon)$ times the minimum contact area of support in any given direction. The expected running time of the algorithm is $O((n/\varepsilon^3)\log^3 n)$.

Our algorithm computes levels in an arrangement of weighted lines (See Section 2 for a precise definition of levels in weighted arrangements). Although much work has been done on computing levels of unweighted lines [1, 8, 10, 11, 13, 6, 7, 16], little is known about computing levels in arrangement of weighted lines. No sub-quadratic bound is known on the complexity of a level in an arrangement of n lines with weights on the plane. In fact an $\Omega(n^{5/3})$ bound is known on the complexity of the median level in weighted arrangements [24]. A recent result by Tamaki *et al.* [17] shows that if the weights of all lines are integers between 1 and U, then the complexity of a level is $O(n(UW)^{1/3})$, where W is the total weight of all the lines. An important result of this paper, which is of independent interest, is an $O((n/\varepsilon^3)\log^3 n)$expected time algorithm for computing a $(1+\varepsilon)$-approximate level, for any given $\varepsilon > 0$. See Section 2 for a precise definition of an approximate level in an arrangement of weighted lines.

For the case of non-convex polyhedron, we show that the set of build directions that minimize the sum of the areas of all the faces that are in contact with support can have as many as $\Omega(n^4)$ connected components. We provide an efficient algorithms for some special case of this problem.

The paper is arranged as follows. In Section 2, we define the levels in weighted arrangements, and present an algorithm that computes an approximate weighted level in an arrangement of lines in the plane. Section 3 describes a near-linear-ime algorithm for finding a build direction that approximately minimizes the contact area of support. Next, in Section 4 we show a lower bound for the number of connected components in the set of build directions that minimize the sum of the areas of the faces in contact with supports for a non-convex polyhedron. Finally, in Section 5 we define a robust build direction, and proceed to describe an algorithm that computes a robust build direction for a non-convex polyhedron.

2 Levels in Arrangements

Let $\mathcal{L} = \{\ell_1, \ell_2, \ldots \ell_n\}$ be a collection of n non-vertical lines in $\mathbb{R}^2$. The *arrangement* $\mathcal{A}(\mathcal{L})$ of $\mathcal{L}$ is the subdivision of $\mathbb{R}^2$ induced by the lines in $\mathcal{L}$. The *level* of a point $p \in \mathbb{R}^2$ with respect to $\mathcal{A}(\mathcal{L})$ is the number of lines in $\mathcal{L}$ that lie strictly below p. The level of all points on an edge of $\mathcal{A}(\mathcal{L})$ remains the same. The *k-level* of $\mathcal{A}(\mathcal{L})$ is the closure of all the edges of $\mathcal{A}(\mathcal{L})$ whose points have level k with respect to $\mathcal{L}$. The k-level of any arrangement is an x-monotone polygonal chain; see Figure 2. There has been much work on the levels of line arrangements. However, the problem of computing the worst case complexity (number of edges) of a k-level in an arrangement of lines on the plane is still open. The best known upper bound is $O(nk^{1/3})$ by Dey [11], and the best known lower bound is $n2^{\Omega(\sqrt{\log k})}$ by Toth [28], which is a recent improvement over a long time lower bound of $\Omega(n \log k)$ [14].

There are several algorithms for computing the k-level in an arrangement of lines. Edelsbrunner and Welzl [13] gave an algorithm for computing the k-level of in an arrangement of lines on the plane, which was later slightly improved by Cole *et al.* [10] to $O(n \log n + b \log^2 k)$ running time, where b is the actual complexity of the k-level. A recent result by Chan [6] improves this bound to $O(n \log n + b \log^{1+\delta} n)$ for any $\delta > 0$. Har-Peled used a different technique to obtain an $O((n+b)\alpha(n)\log n)$ expected time algorithm for the same problem, where $\alpha(n)$ is the inverse Ackermann function. Agarwal *et al.* [1] gave a randomized algorithm for computing the k-level in expected time $O(nk^{1/3} + n \log n)$. A recent analysis by Chan [7] shows that the running time of this algorithm is $O((n+b)\alpha^2(n)\log n)$. The problem of computing approximate levels in arrangement of lines have also been considered by Matousek [22, 23].

In many applications, a positive weight function $w : \mathcal{L} \to \mathbb{R}^+$ is associated with $\mathcal{L}$. In this case we can generalize the definition of level as follows. For a subset $\mathcal{R}$ of $\mathcal{L}$, let $w(\mathcal{R}) = \sum_{\ell \in \mathcal{L}} w(\ell)$. For a point $p \in \mathbb{R}^2$, we define the *lower level*, $lw(p)$ (resp., *upper level,* $uw(p)$), to be the sum of weights of lines that lie strictly below p (resp., that do not lie above p). If p does not lie on any line $\ell \in \mathcal{L}$, then $lw(p) = uw(p)$; if p lies on a line $\ell \in \mathcal{L}$, then $uw(p) = lw(p) + w(\ell)$. All the points on the same edge e of $\mathcal{A}(\mathcal{L})$ have the same lower and upper levels, which we denote by $lw(e)$ and $uw(e)$ respectively.

For a given $\omega \in \mathbb{R}^+$, the *ω-level* in the weighted arrangement $\mathcal{A}(\mathcal{L})$ is the closure of the set of edges e of $\mathcal{A}(\mathcal{L})$ such that $lw(e) \leq \omega < uw(e)$. The ω-level

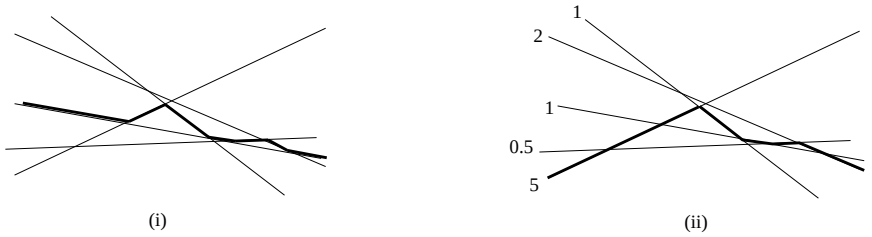

Figure 2. (i) 2-level in arrangement of (i) unweighted lines and (ii)weighted lines.

of a line arrangement is also an x-monotone polygonal chain, but unlike a level in an unweighted arrangement, it may not bend at every vertex it passes through; see Figure 2(ii). If $w(\ell) = 1$, for every $\ell \in \mathcal{L}$, then the definition of a ω-level is identical to the one in the unweighted case. A level in a weighted arrangement can have $\Omega(n^2)$ vertices; see Figure 3. We define a $(\leq \omega)$-*level* to be the closure of the set of all the edges e of $\mathcal{A}(\mathcal{L})$ such that $uw(e) < \omega$.

For a ω-level λ, we define the complexity of λ, denoted by $|\lambda|$, to be the number of vertices at which λ bends, i.e., the vertices at which λ switches from one line to another. As mentioned in the introduction, there exists a set of n lines and a weight function $w : \mathcal{L} \to \mathbb{R}^+$ so that the complexity of the median level in $\mathcal{A}(\mathcal{L})$ is $\Omega(n^{5/3})$ [24]. No subquadratic upper bound is known on the complexity of a level in weighted arrangements. We can extend the algorithms for computing a level in unweighted arrangements to compute a level in weighted arrangements whose running time is proportional to the number of vertices in the level being computed. For example, Har-Peled's algorithm can compute a level λ in a weighted arrangement in expected time $O((n+b)\log n\alpha(n))$, where b is the number of vertices on λ (including those at which λ does not bend). Hence we have the following.

Lemma 2.1 *Let $\mathcal{L}$ be a set of n lines in $\mathbb{R}^2$, $w : \mathcal{L} \to \mathbb{R}^+$ a weight function on $\mathcal{L}$, and value $\omega \geq 0$ a real number. Then the ω-level λ in $\mathcal{A}(\mathcal{L})$ can be computed in expected time $O((n+b)\alpha(n)\log n)$, where b is the number of vertices on the ω-level.*

Since the complexity of a level can be quite large, we consider the problem of computing an approximate level in a weighted arrangement. For a given ω, $0 \leq \omega \leq w(\mathcal{L})$, and a parameter $\varepsilon > 0$, we define a $(1+\varepsilon)$-*approximate ω-level* to be an x-monotone polygonal path Π that lies between the $(1-\varepsilon)\omega$-level and the $(1+\varepsilon)\omega$-level of $\mathcal{A}(\mathcal{L})$. We will be mostly considering approximate levels formed by the edges of $\mathcal{A}(\mathcal{L})$. If an edge e of $\mathcal{A}(\mathcal{L})$ lies on Π, then there exists an $\xi \in [lw(e), uw(e))$ such that $(1-\varepsilon)\omega \leq \xi \leq (1+\varepsilon)\omega$. We define the complexity of Π, denoted by $|\Pi|$, to be the corners of Π. If Π consists of the edges of $\mathcal{A}(\mathcal{L})$, then $|\Pi|$ is the number of vertices of $\mathcal{A}(\mathcal{L})$ at which Π bends.

The problem of computing approximate weighted levels was not previously considered. In this section, we give an $O((n/\varepsilon^3)\log^3 n)$ expected time algorithm for computing the approximate ω-level in an arrangement of lines with weights.

2.1 Computing approximate levels

The weights assigned to the lines can be arbitrary. In this subsection we show that for computing approximate levels we can assume that the weights do not differ by a large factor. Let the maximum and minimum weights among all the lines in $\mathcal{L}$ be $w_{\max}$ and $w_{\min}$ respectively. The following lemma suggests that we can assume $w_{\max}/w_{\min} \leq 4n/\varepsilon$.

Lemma 2.2 *Suppose there is an algorithm that, for any $\varepsilon > 0$, computes in expected time $T(n,\varepsilon)$ a $(1+\varepsilon)$-approximate w-level in an arrangement of n weighted lines provided that $w_{\max}/w_{\min} \leq 4n/\varepsilon$. Let $\mathcal{L}$ be a set of n lines, and let $w : \mathcal{L} \to \mathbb{R}^+$ be an arbitrary weight function. For a given $\omega > 0$, we can compute a $(1+\varepsilon)$-approximate ω-level in $\mathcal{A}(\mathcal{L})$ in time $T(n,\varepsilon/2)$.*

Proof: Discard all the lines $\ell \in \mathcal{L}$ with $w(\ell) < \omega\varepsilon/2n$. For all the lines ℓ with $w(\ell) > 2\omega$, set $w(\ell) = 2\omega$. We are left with a set of n lines such that $w_{\max}/w_{\min} < 4n/\varepsilon$. Hence, we can compute a $(1+\varepsilon/2)$-approximate ω-level in this arrangement in time $T(n,\varepsilon/2)$. The total weight of all the discarded lines is less than $n(\omega\varepsilon/2n) = \omega\varepsilon/2$. Hence, the level that we have computed is a $(1+\varepsilon)$-approximate ω-level. □

In view of the above lemma, we assume that we have a set $\mathcal{L}$ of n lines in $\mathbb{R}^2$ and a weight function $w : \mathcal{L} \to \mathbb{R}^+$ such that $1 \leq w(\ell) \leq 4n/\varepsilon$ for all $\ell \in \mathcal{L}$. We first describe an algorithm for computing a $(1+\varepsilon)$-approximate ω-level under the assumption $1 \leq w(\ell) \leq 2$ for all $\ell \in \mathcal{L}$, and then extend it to the general case.

2.2 Lines with almost equal weights

In this subsection we assume that $1 \leq w(\ell) \leq 2$. For a given $0 \leq w \leq w(\mathcal{L})$ and an $\varepsilon > 0$, we show that a $(1+\varepsilon)$-approximate w-level can be computed in expected time $O((n/\varepsilon)\alpha(n)\log n)$. Since all the weights are between 1 and 2, the following lemma is obvious.

Lemma 2.3 *For any $\omega < w(\mathcal{L}) - 2$, the ω-level and the $(\omega+2)$-level of $\mathcal{A}(\mathcal{L})$ are edge disjoint.*

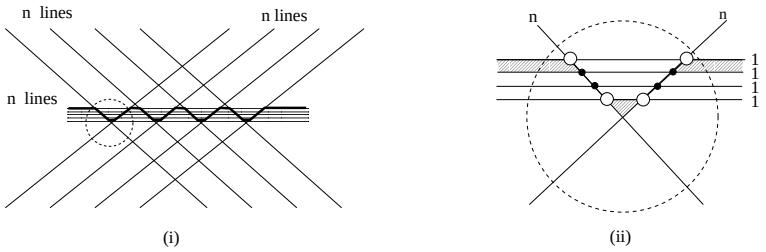

Figure 3. Diagonal lines have weight n and horizontal lines have weight 1. The thick polygonal chain is the (n^2+n-1)-level, which has $\Omega(n^2)$ vertices.

Using the results of Clarkson and Shor [9] (see also Alon and Győri [3]), we can prove the following.

Lemma 2.4 *Let $\mathcal{L}$ be a set of n lines such that $1 \leq w(\ell) \leq 2$ for all $\ell \in \mathcal{L}$. For any real value $\omega < w(\mathcal{L})$, the complexity of the $(\leq \omega)$-level in $\mathcal{A}(\mathcal{L})$ is $O(n\lceil\omega\rceil)$.*

The main result of this subsection is the following lemma.

Lemma 2.5 *Let $\mathcal{L}$ be a set of n lines in $\mathbb{R}^2$ with an associated weight function. Given an $\varepsilon > 0$ and $0 \leq w < w(\mathcal{L})$, there exists a level $\overline{w}$ such that the following three conditions hold: (i) $|w - \overline{w}| \leq \varepsilon w$; (ii) either $w = \overline{w}$, or $\overline{w}$ is an integer; and (iii) the complexity of the $\overline{w}$-level in $\mathcal{A}(\mathcal{L})$ is $O(n/\varepsilon)$.*

Proof: If $w \leq 3/\varepsilon$, set $\overline{w} = w$. By Lemma 2.4, the complexity of the $\overline{w}$-level is $O(n\lceil w\rceil) = O(n/\varepsilon)$. So we can assume that $\varepsilon w > 3$. In this case we choose $\overline{w}$ to be a random integer in the interval $[(1-\varepsilon)w, (1+\varepsilon)w]$. By Lemma 2.3 the levels i and $(i+2)$ are edge disjoint. Since the complexity of the $\leq(1+\varepsilon)w$-level is $O(nw)$, the expected complexity of the $\overline{w}$-level is $O(nw/(2\varepsilon w - 2)) = O(n/\varepsilon)$. This completes the proof of the lemma. □

Using the algorithm of Lemma 2.1, we compute a random x-monotone chain that lies between the $(1-\varepsilon)\omega$-level and the $(1+\varepsilon)\omega$-level. By Lemma 2.5, this has expected complexity $O(n/\varepsilon)$, and can be computed in time $O((n/\varepsilon)\alpha(n)\log(n))$ expected time. Hence, we have the following.

Theorem 2.6 *Given a set $\mathcal{L}$ of n lines, a weight function $w : \mathcal{L} \to \mathbb{R}^+$ such that $1 \leq w(\ell) \leq 2$ for all $\ell \in \mathcal{L}$, two real numbers $\omega, \varepsilon > 0$, we can compute a $(1+\varepsilon)$-approximate ω-level of complexity $O(n/\varepsilon)$ in expected time $O((n/\varepsilon)\alpha(n)\log n)$.*

2.3 Lines with large weights

In this subsection, we assume that $1 \leq w(\ell) \leq 4n/\varepsilon$ for all $\ell \in \mathcal{L}$. We propose a randomized algorithm for computing a $(1+\varepsilon)$-approximate level whose expected running time is $O((n/\varepsilon^3)\log^3 n)$. This result combined with the observations of Lemma 2.2 gives us an algorithm that computes a $(1+\varepsilon)$-approximate level in an arrangement of lines with arbitrary weights.

Lemma 2.7 *Given a set $\mathcal{L}$ of n lines in $\mathbb{R}^2$, a weight function $w : \mathcal{L} \to \mathbb{R}^+$ such that $1 \leq w(\ell) \leq 4n/\varepsilon$, a real value $0 \leq w \leq w(\mathcal{L})$, and a parameter $\varepsilon > 0$, there exists a weighted $\overline{w}$-level λ, in $\mathcal{A}(\mathcal{L})$ such that (i) $|w - \overline{w}| \leq \varepsilon w$ and (ii) $|\lambda| = O((n/\varepsilon^3)\log^3 n)$.*

Proof: Set $\alpha = 4n/\varepsilon$, and $\beta = \varepsilon/4$. Partition lines of $\mathcal{L}$ into buckets $\mathcal{L}_i$ such that a line ℓ is in bucket $\mathcal{L}_i$ if and only if $2^{i-1} \leq w(\ell) < 2^i$. The number of non-empty buckets is at most $\lceil\log_2 \alpha\rceil$; set $n_i = |\mathcal{L}_i|$.

Let $k_{i0} = 0$ and $k_{ij} = 2^{i-1}(1+\beta)^{j-1}$. For each bucket $\mathcal{L}_i$ compute the 0-level λ_{i0} and a $(1+\beta/4)$-approximate k_{ij}-level λ_{ij} of $\mathcal{L}_i$, for $1 \leq j \leq \lceil\log_{1+\beta} n_i\rceil$. By Theorem 2.6, $|\lambda_{ij}| = O(n_i/\beta)$ and λ_{ij} can be computed in expected time $O((n_i/\beta)\alpha(n)\log n)$. Set

$$\Lambda_i = \{\lambda_{ij} \mid 0 \leq j \leq \lceil\log_{1+\beta} n_i\rceil\}.$$

For a point $p \in \mathbb{R}^2$, define $u_i(p)$ to be $k_{i(j+1)}$ if $\lambda_{ij} \in \Lambda_i$ is the polygonal chain lying immediately below p. If p lies on or below λ_{i0}, then $u_i(p) = 0$. Set $u(p) = \sum_{i=1}^{\lceil\log_2 \alpha\rceil} u_i(p)$. It can be shown that $(1-\varepsilon)lw(p) \leq u(p) \leq (1+\varepsilon)uw(p)$ with respect to $\mathcal{A}(\mathcal{L})$.

Let $\Lambda = \bigcup_i \Lambda_i$, and let $\mathcal{A}(\Lambda)$ be the arrangement of all the chains in Λ. Since all the polygonal chains are x-monotone, λ_{ij} and $\lambda_{i'j'}$ intersect in $O(|\lambda_{ij}| + |\lambda_{i'j'}|)$ points. Hence $\mathcal{A}(\Lambda)$ has $O((n/\beta)\log_2\alpha\log_\beta^2\alpha) = O((n/\beta^3)\log^3 n)$ vertices. The weight $u(p)$ of all points p on an edge e of $\mathcal{A}(\Lambda)$ is the same, which we denote by $u(e)$.

For a given $\omega \in \mathbb{R}^+$, we compute the $(1+\varepsilon)$-approximate ω-level (w.r.t $\mathcal{L}$) as follows. Let $e_1, \ldots e_k$ be the set of edges in $\mathcal{A}(\Lambda)$ such that $lu(e_i) \leq \omega$ and $uw(e_i) > \omega$; $e_1, \ldots, e_k$ form a polygonal chain Π. We return the polygonal line Π, which is a $(1+\varepsilon)$ approximate ω-level. □

By Theorem 2.6, each λ_{ij} can be computed in expected time $O((n_i/\varepsilon)\alpha(n_i)\log n_i)$ time. Therefore the total time spent in computing Λ_i is $O((n_i/\varepsilon)\alpha(n)\log n\log_{1+\beta}\alpha) = O((n_i/\varepsilon^2)\alpha(n)\log^2 n)$. Summing over all i, and noting that λ_{ij} and $\lambda_{i'j'}$ intersect in $O(|\lambda_{ij}| + |\lambda_{i'j'}|)$ points, we can compute $\mathcal{A}(\Lambda)$

in expected time $O((n/\varepsilon^3)\log^3 n)$. We finally compute the approximate level by performing a graph search in the 1-skeleton of $\mathcal{A}(\Lambda)$. Hence we have the following.

Theorem 2.8 *Given a set $\mathcal{L}$ of n lines in $\mathbb{R}^2$, a weight function $w : \mathcal{L} \to \mathbb{R}^+$ such that $1 \le w(\ell) \le 4n/\varepsilon$, a real value $0 \le \omega \le w(\mathcal{L})$, and a paramater $\varepsilon > 0$, we can compute a $(1+\varepsilon)$-approximate ω-level λ in $\mathcal{A}(\mathcal{L})$ in expected time $O((n/\varepsilon^3)\log^3 n)$, such that $|\lambda| = O((n/\varepsilon^3)\log^3 n)$.*

Actually, we obtain the following more general result from $\mathcal{A}(\mathcal{L})$:

Theorem 2.9 *Given a set $\mathcal{L}$ of n lines in $\mathbb{R}^2$, a weight function $w : \mathcal{L} \to \mathbb{R}^+$ such that $1 \le w(\ell) \le 4n/\varepsilon$, and a paramater $\varepsilon > 0$, we can preprocess $\mathcal{L}$ in $O((n/\varepsilon^3)\log^3 n)$ expected time so that for a query point p, we can compute in $O(\log n)$ time a real value $\nu(p)$, where $(1-\varepsilon)lw(p) \le \nu(p) \le (1+\varepsilon)lw(p)$. The preprocessing time can be reduced to $O((n/\varepsilon^2)\log^2 n)$ if we allow the query time to be $O(\log^2 n)$.*

2.4 Levels in arrangement of halfplanes

The notion of levels can be extended to a more general setting, which will be useful for our application. Given a family Γ of closed subsets of $\mathbb{R}^2$ (also called *ranges*), and a weight function $w : \Gamma \to \mathbb{R}^+$, define the *level* of a point p, denoted by $\lambda(p)$, to be $\sum_{i \in \mathrm{int}(\gamma), \gamma \in \Gamma} w(\gamma)$ [27]. We call a real value ν a $(1+\varepsilon)$-approximate level of p if $|\nu - \lambda(p)| \le \varepsilon\lambda(p)$. Here we consider the case in which ranges are half planes. Note that the level in an arrangement of halfplanes can have $\Omega(n^2)$ complexity even if the weight of each halfplane is 1; see Figure 4. However, Theorem 2.8 can be modified to construct a $(1+\varepsilon)$-approximate level in $O(n\,\mathrm{polylog}\,n)$ time, even for halfplanes.

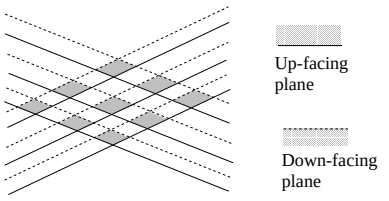

Figure 4. $(\frac{n}{2}+2)$-level has $\Omega(n^2)$ complexity.

We first note that Lemma 2.2 is true for levels in an arrangement of halfplanes as well. Hence, we assume that $w_{\max}/w_{\min} \le 4n/\varepsilon$.

Let Γ^+ (resp., Γ^-) be the set of halfplanes in Γ that face upwards (resp., downwards), i.e., the points in the halfplane lie above (resp., below) the line bounding the halfplane of Γ^+ (resp., Γ^-). Let $\mathcal{L}^+$ and $\mathcal{L}^-$ be the set of lines bounding Γ^+ and Γ^- respectively. A level of a point p in $\mathcal{A}(\Gamma^+)$ is the same as the level of the point in $\mathcal{A}(\mathcal{L}^+)$, and the level of a point in $\mathcal{A}(\Gamma^-)$ is the same as the level of a point in $\mathcal{A}(\mathcal{L}^-)$ if we reverse the direction of the positive y-axis. If the level of a point p in $\mathcal{A}(\Gamma)$ is w, there exists a $w_1 \ge 0$, such that the level p in $\mathcal{A}(\Gamma^+)$ and $\mathcal{A}(\Gamma^-)$ are w_1 and $w - w_1$ respectively. Using this observation, we can extend Theorem 2.8 as follows.

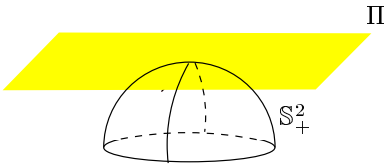

Figure 5. Central Projection.

Theorem 2.10 *Given a collection Γ of n halfplanes in $\mathbb{R}^2$, a weight function $w : \Gamma \to \mathbb{R}^+$, a real value $0 \le w \le w(\Gamma)$, and a parameter $\varepsilon > 0$, a $(1+\varepsilon)$-approximate w-level can be computed in $O((n/\varepsilon^3)\log^3 n)$ time. We can also preprocess Γ in $O((n/\varepsilon^2)\log^2 n)$ time into a data structure so that the $(1+\varepsilon)$-approximate level of a query point can be computed in $O(\log^2 n)$ time.*

3 Minimizing Contact Area of Supports for a Convex Polytope

Let $\mathcal{P}$ be a convex polytope with n faces, and let $\mathbb{S}^2$ be the sphere of directions in $\mathbb{R}^3$. For a face f of $\mathcal{P}$, let $\mathbf{n}_f$ be the outward normal of f. For a given direction $\mathbf{n}$, a face f is a *back face* if $\mathbf{n}_f \cdot \mathbf{n} < 0$. The set of directions for which f is a back face is the open hemisphere $h_f : \mathbf{n}_f \cdot \mathbf{x} < 0$, $\mathbf{x} \in \mathbb{S}^2$. The great circle bounding the hemisphere h_f lies on the plane normal to $\mathbf{n}_f$ and passing through origin (i.e., $\mathbf{n}_f \cdot \mathbf{x} = 0$).

Let $H = \{h_f \subseteq \mathbb{S}^2 \mid f \text{ is a face of } \mathcal{P}\}$ be a set of n hemispheres. For each $h_f \in H$, we define the weight $w(h_f)$ to be the area of the face f. If we build $\mathcal{P}$ in the direction $\mathbf{n}$ using layered manufacturing, then the face f needs support if and only if f is a back face in the direction $\mathbf{n}$. Let $\mu(\mathbf{n})$ denote the total area of all the faces that are in contact with supports for the build direction $\mathbf{n}$. Thus for a convex polyhedron, $\mu(\mathbf{n}) = \sum w(h), h \in H$. Hence, the problem of finding a direction $\mathbf{n}$ that minimizes the total area of support is the same as the problem of finding a direction $\mathbf{n} \in \mathbb{S}^2$ so that $\mu(\mathbf{n})$ is minimized.

Without loss of generality we will assume that none of the great circles bounding h_f is parallel to the xy-plane. Let $\mathbb{S}^2_+$ be the portion the sphere $\mathbb{S}^2$ that lies above the plane $z = 0$, and

$$H_+ = \{h_f^+ = h_f \cap \mathbb{S}^2_+ \mid h_f \in H\}.$$

Let Π be the plane $z = 1$, and let $\pi : H_+ \to \Pi$ be the central projection [25], i.e., for a point $p \in \mathbb{S}^2$, $\pi(p)$ is the intersection point of the ray $\overrightarrow{op}$ with the plane Π; π is a surjective map; see Figure 5.

For a face $f \in \mathcal{P}$, let $\gamma_f = \{\pi(p) \mid p \in h_f^+\}$ be the halfplane lying in Π, which is the central projection of h_f^+. Let $\Gamma = \{\gamma_f \mid f \in \mathcal{P}\}$ be the resulting collection of halfplanes, and let $w : \Gamma \to \mathbb{R}^+$ be the weight function where $w(\gamma_f) = w(h_f)$. For a point $p \in \Pi$, let $\sigma(p) = \sum_{p \in \gamma, \gamma \in \Gamma} w(\gamma)$. Note that for a point $q \in \mathbb{S}_+^2$, $\sigma(\pi(q)) = \sum_{\pi(q) \in \gamma_f, \gamma_f \in \Gamma} w(\gamma_f)$. Therefore, the problem of finding a direction $\mathbf{n}_+ \in H_+$ that minimizes the contact area of support is equivalent to finding a point $p \in \Pi$ that has the minimum weight. We can use a similar procedure to compute a direction $\mathbf{n}_-$ of minimum support area in $\mathbb{S}_-^2$. Between $\mathbf{n}_+$ and $\mathbf{n}_-$, we choose the one for which the contact area is smaller. Using Theorem 2.10, we can compute $\mathbf{n}_+$ in $O((n/\varepsilon^3)\log^3 n)$ time. Hence, we obtain the following.

Theorem 3.1 *Let $\mathcal{P}$ be a convex polytope in $\mathbb{R}^3$ with n faces, and let $\varepsilon > 0$ be a parameter. We can compute in expected time $O((n/\varepsilon^3)\log^3 n)$ a build direction for which the contact area of the support material is at most $(1+\varepsilon)\omega^*$. Here ω^* is the minimum contact area required for building $\mathcal{P}$ in any direction.*

Majhi *et al.* [20] consider the problem of computing a build direction that minimizes the contact area with supports under the condition that $\mathcal{P}$ rests on one of the faces. The problem reduces to computing $\mu(\mathbf{n}_f)$ for every face $f \in \mathcal{P}$. They gave an $O(n^{4/3}\,\mathrm{polylog}\, n)$-time algorithm for this problem. Using Theorem 2.10, we can preprocess $\mathcal{P}$ in $O((n/\varepsilon^2)\log^2 n)$ time into a data structure so that a $(1+\varepsilon)$-approximation of $\mu(\mathbf{n})$ can be computed in $O(\log^2 n)$ time. Hence, we obtain the following result.

Theorem 3.2 *Let $\mathcal{P}$ be a convex polytope in $\mathbb{R}^3$ with n faces, and let $\varepsilon > 0$ be a parameter. We can compute in expected time $O((n/\varepsilon^2)\log^2 n)$ a face f of $\mathcal{P}$ on which $\mathcal{P}$ rests for which the contact area of support is at most $(1+\varepsilon)\omega^*$. Here ω^* is the minimum contact area required for building $\mathcal{P}$ on any face.*

The time required to build $\mathcal{P}$ is proportional to the number of layers required to construct $\mathcal{P}$. Hence, among all the directions that minimize the contact area of support we might want to minimize the number of layers. The number of layers required in the direction $\mathbf{n}$ is proportional to the distance $\delta(\mathbf{n})$ between the two parallel planes that are normal to $\mathbf{n}$ and that support $\mathcal{P}$. By a slight modification of Theorem 3.1 we can compute the set of all directions for which the contact area of support is less than $(1+\varepsilon)\omega^*$. Using this in conjunction with the algorithm of Duncan *et al.* [12] for computing the approximate width, we obtain the following result.

Theorem 3.3 *Given a convex polytope $\mathcal{P}$ and two parameters $A, \varepsilon > 0$, we can compute a build direction $\mathbf{n}$ such that the area of the faces in contact with support is less than $(1+\varepsilon)A$ and the number of layers is less than $(1+\varepsilon)l^*$ in expected time $O((n/\varepsilon^3)\log^3 n)$, where l^* is the minimum number of layers required for all build directions that have the area of contact with supports less than $(1+\varepsilon)A$.*

4 Lower Bound for Non Convex Polyhedra

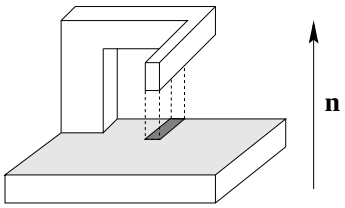

Figure 6. A nonconvex polyhedron with front face requiring support.

In this section we consider the problem of building nonconvex polyhedra using layered manufacturing. Unlike for convex polyhedra, a front face of a nonconvex polyhedra may lie below another face for a given build direction. Therefore, not only the back faces require support, but some of the front faces may also require support (see Figure 6). We thus have the following problem: Given a nonconvex polyhedron $\mathcal{P}$; find a direction so that the total area of the faces that need support is minimized. (Note that even though only a portion of a front face may be occluded, we assume that the whole face is supported.) We will show that developing an $o(n^4)$ algorithm for this problem may be hard.

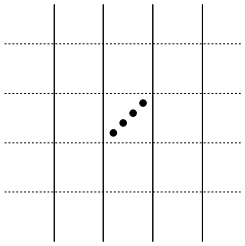

Figure 7. The three main families.

Let $\Delta(\mathcal{P})$ be the set of directions for which the total area of the faces that need support is minimum. We will construct a polyhedron with $O(n)$ faces for which $\Delta(\mathcal{P})$ has $\Omega(n^4)$ connected components. Therefore, any algorithm that checks all the connected components will have $\Omega(n^4)$ running time in the worst case. This problem is related to the so-called *k-sum hard* problem.[1] In the rest of this section, we describe a

[1] A problem is called *k-sum hard* if it can be reduced to the following problem in $o(n^{\lceil k/2 \rceil})$ time [15]: Given a set S of n integers, is the sum of any k items in S equal to 0?

construction of the polyhedron $\mathcal{P}$ and prove the lower bound on $\Delta(\mathcal{P})$.

4.1 Basic idea

We first describe the "main" faces of $\mathcal{P}$ that generate $\Omega(n^4)$ components of $\Delta(\mathcal{P})$. For simplicity we allow the faces of $\mathcal{P}$ to be line segments or points. Later we will connect these faces together and remove the degeneracies. The polyhedron $\mathcal{P}$ consists of three "main" family of faces. The first family is a set $\mathcal{L}_1$ of n line segments parallel to the y-axis and on the plane $z = 2$. The endpoints of the ith segment are $(i/n, -4, 2)$ and $(i/n, 4, 2)$ for $1 \leq i \leq n$. The second family is a set $\mathcal{L}_2$ of n line segments parallel to the x-axis and lying on the plane $z = 1$. The endpoints of the jth segment in $\mathcal{L}_2$ are $(-4, j/n, 1)$ and $(4, j/n, 1)$ for $1 \leq j \leq n$. The third family is a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points on the plane $z = 0$. The coordinates of the point p_i are $(i/(2n^2), i/(2n^2), 0)$; i.e., the points in P lie on the line $y = x$ in the xy-plane; see Figure 7.

Later we will replace each point in P by a triangle of area $1/(2n^2)$, and the lines in $\mathcal{L}_1$ and $\mathcal{L}_2$ by skinny triangles with area less than $1/n^8$. So only P will have a non-zero area, and any point (or triangle) in P will not need support for any direction in $\Delta(\mathcal{P})$. We will show that the set of directions in $\mathbb{S}^2$ for which no point of P needs support because of $\mathcal{L}_1 \cup \mathcal{L}_2$ consists of $\Omega(n^4)$ components. These components will remain disconnected even after we connect the three families together to form a polyhedron. We will now prove a bound on the number of connected components of this set.

4.2 Lower bound

A point p will need support because of line segment ℓ in the build direction $\mathbf{n}$ if the ray $p + t\mathbf{n}$, $t \geq 0$, intersects ℓ. For a point p and a segment ℓ, let $C(p, \ell)$ denote the set of directions on $\mathbb{S}^2$ for which the point p requires support because of the segment ℓ. $C(p, \ell)$ is a portion of a great semicircle; the great circle containing $C(p, \ell)$ is the intersection of $\mathbb{S}^2$ with the plane passing through the origin and parallel to the plane containing both p and ℓ (See Figure 8(i)). Set $\mathcal{C}_1 = \{C(p, \ell \mid p \in P, \ell \in \mathcal{L}_1\}$ and $\mathcal{C}_2 = \{C(p, \ell \mid p \in P, \ell \in \mathcal{L}_2\}$.

Let $\Sigma \subseteq \mathbb{S}^2$ be the spherical cap of angular opening $\pi/3$ centered at the north pole of $\mathbb{S}^2$. We will show that $\Delta(\mathcal{P})$ has $\Omega(n^4)$ connected components inside Σ. It can be checked that, for all $p \in P$, $\ell \in \mathcal{L}_1 \cup \mathcal{L}_2$, $C(p, \ell)$ intersects Σ, and that its endpoints lie outside Σ. The following lemma is easy to observe.

Lemma 4.1 *Let ℓ be a line parallel to the y-axis. Then for any point $p \in \mathbb{R}^3$, $C(p, \ell)$ is a great semicircle whose endpoints are $(0, \pm 1, 0)$*

The above lemma implies that for any two points $p, p' \in \mathbb{R}^3$ and two lines ℓ, ℓ' parallel to the y-axis, if

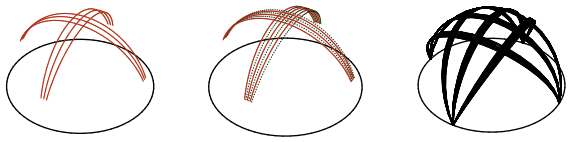

Figure 8. (i)$C(p, \ell)$ for a point and n lines; (ii) $C(p, \ell)$ for n points and n lines; (iii) $C(t_p, \ell)$ for triangles and lines.

$C(p, l)$ and $C(p', \ell')$ are distinct, then their only intersection points are $(0, \pm 1, 0)$. The preceding discussion implies the following:

Lemma 4.2 *Let $p = (p_x, p_y, 0)$ and $p' = (p'_x, p'_y, 0)$ be two points on the xy-plane, and let $\ell : x = i$, and $\ell' : x = i'$ be two lines parallel to the y axis lying on the same horizontal plane. Then $C(p, \ell) = C(p', \ell')$ if and only if $i - p_x = i' - p'_x$.*

Similar properties can also be proved for lines parallel to x-axis. The choice of the parameters for P and for $\mathcal{L}_1$ and $\mathcal{L}_2$ imply that all the arcs in $\mathcal{C}_1$ and $\mathcal{C}_2$ are distinct. Moreover, the following lemma, whose proof is omitted from here, implies that all pairs of arcs in $\mathcal{C}_1 \times \mathcal{C}_2$ intersect inside the cap Σ.

Lemma 4.3 *For any two pairs, $(p, \ell_1) \in P \times \mathcal{L}_1$ and $(p', \ell_2) \in P \times \mathcal{L}_2$, the arcs $C(p, \ell_1)$ and the arc $C(p', \ell_2)$ intersect inside the spherical cap Σ.*

Hence, we have an $n^2 \times n^2$ grid of circular arcs within the spherical cap Σ; see Figure 8(ii). This shows that the set of directions for which no point in P is obstructed by a line of $\mathcal{L}_1 \cup \mathcal{L}_2$ can have $\Omega(n^4)$ connected components. Next we show how to form a polyhedron out of P, $\mathcal{L}_1$, and $\mathcal{L}_2$ and argue that $\Delta(\mathcal{P})$ has $\Omega(n^4)$ connected components even after the last step.

4.3 Constructing the polyhedron

We construct the polyhedron in two stages. In the first stage we replace each point $p \in P$ with a triangle τ, and then connect all the parts to construct the polyhedron.

Replace each point $p \in P$ with an equilateral triangle τ_p of side length $1/(8n^2)$ such that p is the center of τ_p. Let T be the set of resulting triangles. For a triangle τ_p and a line $\ell \in \mathcal{L}_1 \cup \mathcal{L}_2$, define $C(\tau_p, \ell)$ to be the set of directions in $\mathbb{S}^2$ such that the triangle τ_p will require support because of the line ℓ. We can prove the following lemma.

Lemma 4.4 *The set of directions for which none of the triangles τ_p, $p \in P$ need support due to the lines in $\mathcal{L}_1 \cup \mathcal{L}_2$ has $\Omega(n^4)$ connected components.*

For constructing the polyhedron from $\mathcal{L}_1$, $\mathcal{L}_2$ and T, we attach the line segments in the set $\mathcal{L}_1$ with another line segment whose endpoints are $(-4, -4, 2)$ and $(-4, 4, 2)$. Similarly we attach the line segments in $\mathcal{L}_2$ by adding a new segment with endpoints $(-4, -4, 1)$,

and $(4, -4, 1)$. We now join the two points $(-4, -4, 2)$ and $(-4, -4, 1)$ and extend it till $z = -1$, thus having a segment with end points $(-4, -4, 2)$ and $(-4, -4, -1)$. We add another segment along the line $y = x$ in the plane $z = -1$, and erect thin pillars from this segment which support the triangles τ_p. By appropriately expanding each segment to a very thin rectangle (say of area $< 1/n^8$), we can ensure that the same set of new faces need support for all directions within the spherical cap Σ. Hence, each connected component mentioned in the previous section is still a connected component. Omitting all the details, we obtain the following.

Theorem 4.5 *There exists a polyhedron $\mathcal{P}$ in $\mathbb{R}^3$ with n faces for which $\Delta(\mathcal{P})$ has $\Omega(n^4)$ connected components.*

5 Minimizing Support for Non-convex Polyhedra

As shown in the previous section, the problem of minimizing the contact area of supports seems to be a rather difficult problem for nonconvex polyhedra. Therefore, we consider interesting variants of the problem, which are relevant for some real practical models on nonconvex polyhedron. In this section, we first present an efficient algorithm for determining area of contact support. Next we consider the problem of a "robust" build direction.

5.1 Computing the Contact Area

Let $\mathcal{P}$ be a non-convex polyhedron with n faces and $\mathbf{n}$ a build direction. Recall that $\mu(\mathbf{n})$ denotes the total area of all the faces that are in contact with support for the build direction $\mathbf{n}$. In this subsection, we give algorithms for computing $\mu(\mathbf{n})$ for a given direction $\mathbf{n}$ and polyhedron $\mathcal{P}$.

An efficient algorithm. We first determine in $O(n)$ time all the back faces and compute their area. Let Δ be the set of front faces of $\mathcal{P}$ in the direction $\mathbf{n}$. A face f needs support if another triangle of Δ is occluding a point in f. The problem thus reduces to determining the subset of triangles in Δ that are partially or totally occluded. Without loss of generality, assume that $\mathbf{n}$ is the $(+z)$-direction. A triangle $f \in \Delta$ is partially occluded if at least one of the following conditions holds:

(i) A vertical ray emanating from a vertex of f intersects another triangle of Δ.

(ii) For an edge $e \in f$, the *curtain*

$$\overline{e} = \{(x, y) \times [z, \infty) \mid (x, y, z) \in e\}$$

intersects the edge of another face in Δ.

(iii) The interior of the semi-infinite vertical prism

$$\overline{f} = \{(x, y) \times [z, \infty) \mid (x, y, z) \in f\}$$

contains a vertex of a face in Δ.

Agarwal and Matousek [2] have shown that a set of m triangles in $\mathbb{R}^3$ can be preprocessed in $O(m^{4/3} \log^c m)$ time so that a vertical ray shooting or a curtain query can be answered in time $O(m^{1/3} \log^c m)$. Given a set of m points in $\mathbb{R}^3$, one can combine the two-dimensional simplex range searching data structure (using partition trees) with a three-dimensional half plane range reporting data structure in time $O(m^{4/3} \log^c m)$ so that a query of type (iii) can be answered $O(m^{1/3} \log^c m)$ time. Hence, after $O(n^{4/3} \log^c n)$ preprocessing, we can determine in $O(n^{4/3} \log^c n)$ time whether a triangle of Δ is partially occluded. We thus obtain the following result.

Theorem 5.1 *Given a non-convex polyhedron $\mathcal{P}$ and a build direction $\mathbf{n}$, we can determine in $O(n^{4/3} \log^c n)$ time, for some constant $c > 0$, the area of faces that need support.*

A simpler algorithm. We next describe a simpler algorithm for computing $\mu(\mathbf{n})$. In the worst case, its running time is quadratic, but in practice it will be much smaller.

For a direction $\mathbf{n}$, an edge e of the polyhedron $\mathcal{P}$ is called a *contour edge* if it is adjacent to both a front face and a back face of $\mathcal{P}$. In the worst case there are $\Omega(n)$ contour edges, but for a typical CAD model the number of contour edges is small compared to the total number of edges in the polyhedron [4, 18]. For example, Kettner and Welzl [18] show that for convex approximation of spheres, the expected number of contour edges for a random viewing direction is only $O(\sqrt{n})$.

We again assume that $\mathbf{n}$ is the $(+z)$-direction. Let $\mathcal{C}$ be the collection of all contour edges. They form disjoint cycles. The cycles partition the front faces of the polyhedron into connected regions possibly with holes, called *contour regions*. Consider the projection of the contour cycles and of the regions on the xy-plane. For each face ψ of the arrangement, identify the contour region that lies on the top, and find the set P_ψ of faces of $\mathcal{P}$ in this contour region whose projections lie entirely inside ψ. The faces in P_ψ do not need support. Repeating this for all the faces of the arrangement of contour edges, we can compute the faces of P that do not need support. After having computed the arrangement, the identification of the faces can be done in $O(n)$ time by traversing ∂P. If k is the number of contour edges and $I \leq \binom{k}{2}$ is the number of intersection points of the projections of the contour edges, then the total running time of this algorithm is $O(k_{\mathbf{n}} \log n + I_{\mathbf{n}} + n)$. Hence we have the following.

Theorem 5.2 *Given a nonconvex polyhedron $\mathcal{P}$ and a direction $\mathbf{n}$, we can compute $\mu(\mathbf{n})$ in time $O(n + k_{\mathbf{n}} \log n + I_{\mathbf{n}})$, where $k_{\mathbf{n}}$ is the number of contour edges of $\mathcal{P}$ and $I_{\mathbf{n}}$ is the number of intersection points in the projections of the contour edges on a plane orthogonal to $\mathbf{n}$.*

5.2 Finding a robust direction

For a direction $\mathbf{n}$, let $D(\mathbf{n}, \delta)$ denote the spherical cap of angular opening δ, centered at $\mathbf{n}$. We call a direction δ-robust if for all directions in $D(\mathbf{n}, \delta)$, the set of faces that need support is the same as for $\mathbf{n}$. Let $A(\delta)$ be the minimum area of contact needed for any δ-robust direction.

Partition $\mathbb{S}^2$ into a grid Q of latitudes and longitudes so that the diameter of each grid cell is at most 2δ. Let $\mathcal{N}$ be the set of directions corresponding to the centers of the grid cells in Q. It is easily seen that $\min_{\mathbf{n} \in \mathcal{N}} \mu(\mathbf{n}) \leq A(\delta)$. By Theorem 5.1 we can compute $\mu(\mathbf{n})$ in time $O(n^{4/3} \log^c n)$ time. Hence, we obtain the following.

Corollary 5.3 *Given a nonconvex polyhedron $\mathcal{P}$ with n vertices and a parameter $\delta > 0$, we can compute in time $O((n^{4/3}/\delta^2) \log^c n)$ a build direction $\mathbf{n}$ such that $\mu(\mathbf{n}) \leq A(\delta)$.*

A problem with this approach is that the direction $\mathbf{n}$ computed might not be δ-robust. If we want to compute a δ-robust direction, we proceed as follows.

Let $\mathcal{N}' \subseteq \mathcal{N}$ be the subset of directions $\mathbf{n}$ such that for all directions in the grid cell of Q centered at $\mathbf{n} \in \mathcal{N}'$, the same set of faces of $\mathcal{P}$ need support. We first compute $\mathcal{N}' \subseteq \mathcal{N}$ and then compute $\mu^* = \min_{\mathbf{n} \in \mathcal{N}'} \mu(\mathbf{n})$ using Theorem 5.1. We note that if a direction $\mathbf{n}$ is 4δ-robust then $D(\mathbf{n}, 4\delta)$ contains a grid cell of Q whose center is δ-robust. Therefore we have that $\mu^* \leq A(4\delta)$. We now describe the procedure to compute $\mathcal{N}'$.

Consider a face f of the polyhedron $\mathcal{P}$. Let $C(f)$ be the set of directions on $\mathbb{S}^2$ for which the face f will require support. The set $C(f)$ is composed of two parts — the first part corresponds to the the set of directions for which f is a back face, which is a hemisphere. The second part consists of a set of directions for which the face f is occluded by another face of $\mathcal{P}$. This second part is the union of $O(n)$ spherical polygons, each composed of $O(1)$ great-circle arcs. It thus suffices to check which of the grid cells do not intersect $\partial C(f)$ for all $f \in \mathcal{P}$, and we choose the cell for which the area of contact is minimum.

Suppose $|\partial C(f)| = m_f$. We can preprocess $\partial C(f)$ using $O(m_f \log m_f)$ space, so that we can determine in $O(\log n)$ time whether a grid cell τ intersects $\partial C(f)$. We repeat the above procedure for all the cells of Q. Let K be the total time spent in computing $\mathcal{N}'$. In the worst case, $K = O(n^3 + (n/\delta^2) \log n)$ but in practice it will be close to quadratic. We can also compute $\mathcal{N}'$ using a quad tree. We construct a quad tree T on Q, i.e., the leaves of T correspond to the grid Q. We store the spherical polygons in T in the standard manner. We omit the details from here. After having computed $\mathcal{N}'$ we can compute $\mu(\mathbf{n})$ either by Theorem 5.1 or by Theorem 5.2. We thus obtain the following result.

Theorem 5.4 *Let $\mathcal{P}$ be a nonconvex polyhedron with n faces. Suppose the set $\mathcal{N}'$ can be computed in $O(K)$ time, then we can compute in time $O(K + (n^{4/3}/\delta^2) \log^c n)$, or in time $O(K + (n/\delta^2) + (1/\delta^2) \sum_{\mathbf{n} \in \mathcal{N}'} (k_{\mathbf{n}} \log n + I_{\mathbf{n}}))$, a δ-robust build direction $\mathbf{n}$ for which $\mu(\mathbf{n}) \leq A(4\delta)$. Here $k_{\mathbf{n}}$ is the number of contour edges in the direction $\mathbf{n}$, and $I_{\mathbf{n}}$ is the number of intersections of the contour edges when they are projected on a plane orthogonal to $\mathbf{n}$.*

References

[1] P. K. Agarwal, M. de Berg, J. Matoušek, and O. Schwarzkopf, Constructing levels in arrangements and higher order Voronoi diagrams, *SIAM J. Comput.*, 27 (1998), 654–667.

[2] P. K. Agarwal and J. Matoušek, Ray shooting and parametric search, *SIAM J. Comput.*, 22 (1993), 794–806.

[3] N. Alon and E. Győri, The number of small semispaces of a finite set of points in the plane, *J. Combin. Theory Ser. A*, 41 (1986), 154–157.

[4] A. Appel, The notion of quantitative invisibility and the machine rendering of solids, *Proc. ACM National Conference*, 1967, pp. 367–393.

[5] B. Asberg, G. Blanco, P. Bose, J. Garcia-Lopez, M. Overmars, G. Toussaint, G. Wilfong, and B. Zhu, Feasibility of design in stereolithography, *Algorithmica*, 19 (1997), 61–83.

[6] T. Chan, Dynamic planar convex hull operations in near-logarithmic amortized time, *Proc. 40th Annu. IEEE Sympos. Found. Comput. Sci.*, 1999, pp. 92–99.

[7] T. Chan, Remarks on k-level algorithms in the plane, *Manuscript*, 1999.

[8] T. M. Chan, Random sampling, halfspace range reporting, and construction of $(\leq k)$-levels in three dimensions, *Proc. 39th Annu. IEEE Sympos. Found. Comput. Sci.*, 1998, pp. 586–595.

[9] K. L. Clarkson and P. W. Shor, Applications of random sampling in computational geometry, II, *Discrete Comput. Geom.*, 4 (1989), 387–421.

[10] R. Cole, M. Sharir, and C. K. Yap, On k-hulls and related problems, *SIAM J. Comput.*, 16 (1987), 61–77.

[11] T. K. Dey, Improved bounds on planar k-sets and related problems, *Discrete Comput. Geom.*, 19 (1998), 373–382.

[12] C. A. Duncan, M. T. Goodrich, and E. A. Ramos, Efficient approximation and optimization algorithms for computational metrology, *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, 1997, pp. 121–130.

[13] H. Edelsbrunner and E. Welzl, Constructing belts in two-dimensional arrangements with applications, *SIAM J. Comput.*, 15 (1986), 271–284.

[14] P. Erdős, L. Lovász, A. Simmons, and E. Straus, Dissection graphs of planar point sets, in: *A Survey of Combinatorial Theory* (J. N. Srivastava, ed.), North-Holland, Amsterdam, Netherlands, 1973, pp. 139–154.

[15] A. Gajentaan and M. H. Overmars, n^2-hard problems in computational geometry, Report RUU-CS-93-15, Dept. Comput. Sci., Utrecht Univ., Utrecht, Netherlands, April 1993.

[16] S. Har-Peled, Taking a walk in a planar arrangement, *Proc. 40th Annu. IEEE Sympos. Found. Comput. Sci.*, 1999, pp. 100–110.

[17] N. Katoh, H. Tamaki, and T. Tokuyama, Parametric polymatroid optimization and its geometric applications, *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, 1999, pp. 517–526.

[18] L. Kettner and E. Welzl, Contour edge analysis for polyhedral projections, Report 258, Departement Informatik, ETH, Zurich, Switzerland, 1996.

[19] J. Majhi, R. Janardan, M. Smid, and P. Gupta, On some geometric optimization problems in layered manufacturing, *Proc. 5th Workshop Algorithms Data Struct.*, *Lecture Notes Comput. Sci.*, Vol. 1272, Springer-Verlag, 1997, pp. 136–149.

[20] J. Majhi, R. Janardan, M. Smid, and P. Gupta, On some geometric optimization problems in layered manufacturing, Report 2, Department of Computer Science, University of Magdeburg, Magdeburg, Germany, 1998.

[21] J. Majhi, R. Janardan, M. Smid, and J. Schwerdt, Multi-criteria geometric optimization problems in layered manufacturing, *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, 1998, pp. 19–28.

[22] J. Matoušek, Construction of ϵ-nets, *Discrete Comput. Geom.*, 5 (1990), 427–448.

[23] J. Matoušek, Approximate levels in line arrangements, *SIAM J. Comput.*, 20 (1991), 222–227.

[24] J. Matoušek, Lower bounds on the length of monotone paths in arrangements, *Discrete Comput. Geom.*, 6 (1991), 129–134.

[25] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, NY, 1985.

[26] J. Schwerdt, M. Smid, J. Majhi, and R. Janardan, Computing the width of a three-dimensional point set: an experimental study, Report 18, Department of Computer Science, University of Magdeburg, Magdeburg, Germany, 1998.

[27] M. Sharir, On k-sets in arrangements of curves and surfaces, *Discrete Comput. Geom.*, 6 (1991), 593–613.

[28] G. Toth, Point sets with many k-sets, *Manuscript*, (1999).