Freeform Shape Machining Using Minkowski Operations

J.W.H. Tangelder,
J.S.M. Vergeest,
and M.H. Overmars

UU-CS-1996-24
June 1996

ISSN: 0924-3275

Freeform Shape Machining Using Minkowski Operations

Johan W.H. Tangelder, Joris S.M. Vergeest
Faculty of Industrial Design Engineering
Delft University of Technology
Jaffalaan 9, NL-2628 BX Delft, The Netherlands
Phone +31 15 2781061, Fax +31 15 2787316
j.w.h.tangelder@io.tudelft.nl, j.s.m.vergeest@io.tudelft.nl

Mark H. Overmars*
Department of Computer Science
Utrecht University
P.O. Box 80089, NL-3508 TB Utrecht, The Netherlands
Phone +31 30 2533736, Fax +31 30 2513791
markov@cs.ruu.nl

Abstract

In this paper we use Minkowski operations to describe freeform shape machining algorithms. Given a cutting tool, a toolholder and a stock-in-progress, that encloses the model to be machined, the computation of a tool path, for which the tool does not interfere the model and the toolholder does not interfere the stock-in-progress is described using the Minkowski addition. The computation of the stock-in-progress that is left, if the tool has followed the tool path, is described using the Minkowski subtraction. Grids of height values are described by real-valued functions on finite subsets of $\mathbb{Z}^2$, called numerical functions. We use Minkowski operations on these numerical functions to describe the well-know "remove as much material as possible" machining strategy and the well-know "slicing" machining strategy. As far as we know these strategies have not been described using Minkowski operations. Since the "slicing strategy" generates tool paths that are machined faster, an efficient implementation of the "slicing strategy" is described using Z-pyramids.

Keywords: Minkowski operations, freeform shapes, interference avoidance, rapid prototyping, Z-buffer machining, Z-pyramids.

*This author was partially supported by the Netherlands Organization for Scientific Research (N.W.O.)

1 Introduction

The freeform shape machining problem includes both an interference avoidance problem and a volume processing problem. Given a cutting tool, a toolholder and a stock-in-progress, that encloses the model to be machined, only tool paths for which the tool does not interfere the model and the toolholder does not interfere the stock-in-progress, are allowed. The stock-in-progress that is left, if the tool has followed the tool path, is the result of a volume removal process by the tool from the stock-in-progress. We show that both this interference avoidance problem and this volume processing problem can be described using Minkowski operations.

In this paper we consider only removal of material by the cutting tool, whereas faster plans might be found by cutting off large chunks (rather than machining them away) in some cases. We also do not consider the stresses of possible breakage of the desired shape (such as a very thin wall, etc.) due to cutting forces.

For the freeform shape machining task we have installed a Sculpturing Robot system consisting of an industrial Manutec R15 robot with 6 rotational degrees of freedom and a turntable that can rotate around its z-axis, see figure 1. The robot holds a milling tool and a stock of foam is mounted on the turntable. The machining pro-

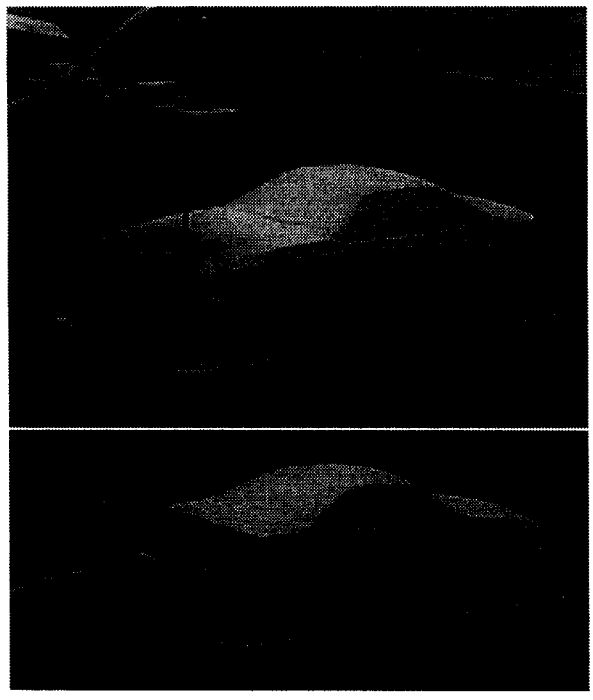

Figure 1: The Sculpturing Robot system performs rapid prototyping of CAD-defined objects. The machining process consists of a number of stages. At each stage the milling tool approaches the upper face or one of the four side faces of the foam stock.

cess shrinks the initial foam stock to a foam prototype of a CAD model. The machining process must be performed in such a way that a subsequent, more accurate machining process can be made. Hence, the obtained prototype should always enclose the freeform shape F. Previously we have developed and implemented the SRPLAN1 robot motion planning and foam machining algorithm that uses only 5 predefined tool orientations [14]. Currently, we investigate methods to extend the shape domain and the size of the prototypes. These methods will be implemented in a new motion planner, that derives a limited number of suitable tool access directions from the CAD model geometry. For each tool access direction a robust and efficient freeform shape machining strategy will be implemented. In this paper we describe such strategies using Minkowski operations.

Researchers in the field of mathematical morphology have studied the Minkowski addition and subtraction extensively [5, 13]. In this field the Minkowski addition and subtraction are used to define the dilation and erosion image processing operations, respectively. Although mathematical morphology has initially been applied to digital image processing, the Minkowski addition and subtraction are defined generally for n-dimensional space. Therefore, with the Minkowski operations also volume processing can be described. E.g. blending solids using Minkowski operations has been described [11].

Also, the Minkowski addition and subtraction has been applied [4, 10] in the spatial planning problem. The spatial planning problem involves placing an object among other objects or move it without colliding with nearby objects. If the object can translate but not rotate the Minkowski addition and subtraction are used to compute the forbidden region and the feasible region, respectively. Avoiding interference between the tool and the model and between the toolholder and the stock-in-progress is a spatial planning problem.

In section 2 we formally define Minkowski operations on sets. In section 3 we describe the freeform shape machining problem using Minkowski operations on sets for a milling tool with a fixed orientation. Next, we focus on the description of well-know free-form shape machining algorithms that represent the tool, the toolholder, the stock-in-progress and the model with grids of height values. In section 4 we formally define these grids of height values as numerical functions. We define Minkowski operations on these numerical functions. In section 5 we use Minkowski operations on numerical functions to describe the well-know "remove as much material as possible strategy" machining strategy (see e.g. [14], [18]) and the well-know "slicing" machining strategy (see e.g. [12]). As far as we know these strategies have not been described using Minkowski operations. [7] and [16] describe the relation between Minkowski operations and NC machining, but no toolholder is included in their analysis and no machining strategies are described. In section 6 we optimize the running time of the computation of the tool paths with the "slicing strategy" using Z-pyramids. We conclude in section 7.

2 Minkowski operations on sets

We have found in the literature several conventions to define Minkowski operations. We adopt the convention used by Serra [13].

The Minkowski addition of two sets A and B, each consisting of vectors in $\mathbb{R}^n$, the Euclidean space of dimension n, is constructed by translating A by each element of B and then taking the union of all the resulting translates, as illustrated in figure 2. Intuitively, the Minkowski addition can be considered as a dilation process where one set is expanded by the other. In more formal terms, the **Minkowski addition** is defined as

$$A \oplus B = \cup_{\mathbf{b} \in B} A + \mathbf{b},$$

where $A + \mathbf{b} = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A\}$ denotes a translation of A by a vector $\mathbf{b}$. The Minkowski addition is commutative, i.e.,

$$A \oplus B = B \oplus A.$$

The Minkowski subtraction of two sets A and B is constructed by translating A by each element of B and then taking the intersection of all the resulting translates as illustrated in figure 3. Intuitively, the Minkowski subtraction can be considered as an erosion process where one set is eroded by the other. In more formal terms, the **Minkowski subtraction** is defined as

$$A \ominus B = \cap_{\mathbf{b} \in B} A + \mathbf{b}.$$

Let c denote set complementation. The Minkowski addition and subtraction are dual operations, i.e., dilating (eroding) a set A by a set B is equivalent with eroding (dilating) A^c by B. In more formal terms,

$$A \oplus B = (A^c \ominus B)^c$$

and

$$A \ominus B = (A^c \oplus B)^c.$$

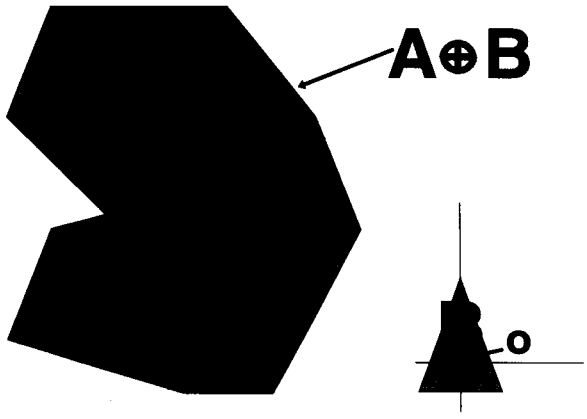

Figure 2: The Minkowski addition $A \oplus B$ is constructed by translating A by each element of B and then taking the union of all the resulting translates. The triangles denote translates of the vertices of A.

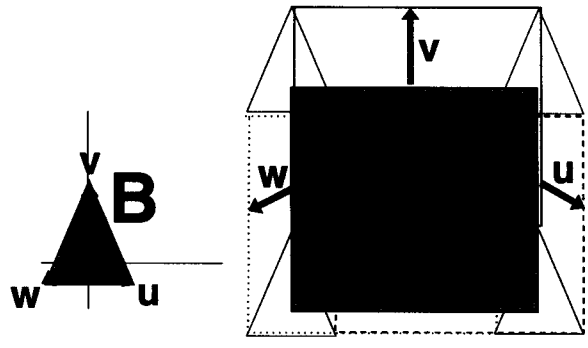

Figure 3: The Minkowski subtraction $A \ominus B$ is constructed by translating A by each element of B and then taking the intersection of all the resulting translates.

3 Freeform shape machining with a fixed tool orientation

We assume that a prototype $R \subset \mathbb{R}^3$, that encloses and approximates a freeform shape F, is to be obtained from a stock of material. The stock-in-progress S is a volume that shrinks from the initial stock to the prototype R due to the penetrating milling tool T. Collision between the toolholder H and S is not allowed. With these assumptions we state the freeform shape machining problem as follows.

Description of freeform shape machining problem

Let a freeform shape F and a stock-in-progress S be given in the same Cartesian coordinate system $\mathcal{S}$. Let the tool T and its toolholder H be given in

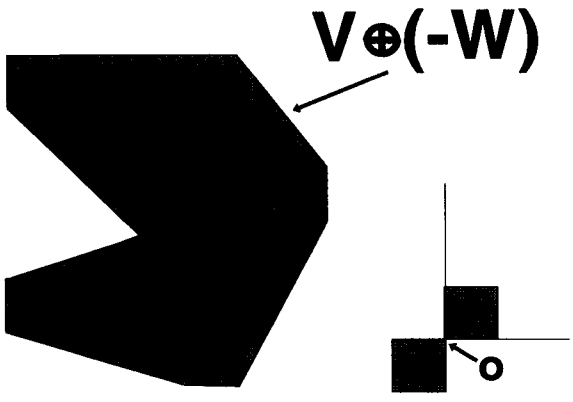

Figure 4: V and W intersect, if $\mathbf{o} \in V \oplus (-W)$.

the Cartesian coordinate system $\mathcal{T}$. Assume that T contains the origin $\mathbf{o}$ of $\mathcal{T}$. Assume that $T \cup H$ is a free-translating object in $\mathcal{S}$, i.e., $\mathcal{T}$ can translate but $\mathcal{T}$ can not rotate in $\mathcal{S}$. Further, assume that the orientations of the x, y and z-axes are identical in $\mathcal{S}$ and $\mathcal{T}$.

Find a toolpath surface P for machining a prototype R of F, such that R is the minimal enclosing volume of F that can be obtained without interference between T and F and without interference between H and S, if the tool follows the toolpath surface.

End of description

In practice a toolpath will be extracted from the toolpath surface P with a certain accuracy. The toolpath is a curve on P such that the machined surface obtained by following the toolpath approximates the toolpath surface within that accuracy.

For a volume $V \subset \mathbb{R}^3$ let $-V = \{-v \mid v \in V\}$ be the symmetrical set of V. The following lemma describes the result of the freeform shape machining process if we neglect interference between the stock-in-progress S and the toolholder H.

Lemma

If interference between T and F is taken into account, but not the interference between S and H, then the final result of the milling process is $(F \oplus (-T)) \ominus T$.

Proof

Sweeping volumes and forbidden volumes for positioning the origin $\mathbf{o}$ of $\mathcal{T}$ can be characterized with help of the Minkowski addition. Let a volume V be given in $\mathcal{S}$ and a volume W be given in $\mathcal{T}$. If $\mathbf{o}$ follows P the volume swept by W is

$$\cup_{\mathbf{p} \in P} W + \mathbf{p} = W \oplus P = P \oplus W.$$

In robot motion planning (see, e.g. [8, 10]), it is a well-know fact that V interferes with W, if $\mathbf{o}$ is positioned such that

$$\mathbf{o} \in \{\mathbf{p} \mid V \cap (W + \mathbf{p})$$

This is the forbidden region for $\mathbf{o}$ in $\mathcal{S}$ if collision between V and W should be avoided (see also figure 4). Hence, interference between the tool T and the shape F occurs if $\mathbf{o} \in F \oplus (-T)$, and interference between the stock-in-progress S and the toolholder H occurs if $\mathbf{o} \in S \oplus (-H)$.

If W contains $\mathbf{o}$, then $V \ominus W = (V^c \oplus W)^c \subseteq V$ is the volume that is left of V after $\mathbf{o}$ has been positioned at every $\mathbf{p}$ outside V, while the volume swept by W, i.e., $V^c \oplus W$ is removed from V. If W does not contain $\mathbf{o}$ then $V \ominus W \subseteq V$ may be violated and $V \ominus W$ does not model the volume that is left of V. Since T contains the origin $\mathbf{o}$ of $\mathcal{T}$ the minimal volume that can be obtained with T and encloses F is $(F \oplus (-T)) \ominus T$, if we neglect interference between S and H.

End of proof

In the field of mathematical morphology $(F \oplus (-T)) \ominus T$ is called the closing of F with respect to T. If T is a sphere centered at $\mathbf{o}$ with radius α then $(F \oplus (-T)) \ominus T$ is identical to the "filleted" blend of F [11]. It is also identical to the α-hull of F, as introduced by Edelsbrunner [2].

If we do not neglect interference between S and H the optimal result of the milling process can be specified as follows.

Description of the result of the milling process

The optimal result R of the milling process is described iteratively as follows. Let $S_0 \supseteq F$ denote the initial stock-in-progress. Let S_i denote the stock-in-progress after step i. S_{i+1} can be obtained from S_i by positioning $\mathbf{o}$ anywhere outside S_i without interference between F and T or between S_i and H. Hence, $S_{i+1} = S_i \cap (P_i \ominus T)$, where P_i is the boundary of $(F \oplus (-T)) \cup (S_i \oplus (-H))$. Since $\forall i : F \subseteq S_{i+1} \subseteq S_i$, $\lim_{i \to \infty} S_i$ exists and $F \subseteq \lim_{i \to \infty} S_i$. Hence, we define R as follows. Let $R = S_i$ if there exists an i such that $S_i = S_{i+1}$. Otherwise let $R = \lim_{i \to \infty} S_i$. Figure 5 illustrates the first step of this description.

End of description

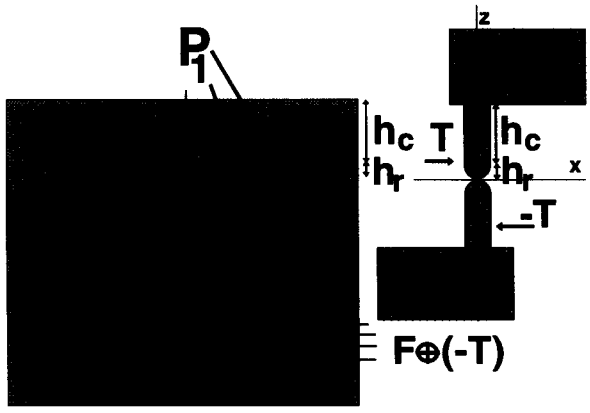

Figure 5: First step of the milling process, where P_1 is the boundary of $(F \oplus (-T)) \cup (S_0 \oplus (-H))$ and $S_1 = S_0 \cap (P_1 \ominus T)$.

This "remove as much material as possible" strategy has been implemented without using Minkowski operations (see, e.g., [14, 18]).

From the S_i in this description, an interference free toolpath can be derived, but a shorter interference free toolpath can be extracted from a tool path surface P, derived directly from R as follows. We restrict ourselves to cutting tools that consist of a cylindric part with height h_c and a rotation symmetric remaining part with height h_r. In the case of a ball end tool the remaining part is a hemisphere with radius h_r and in the case of a flat end tool the remaining part is empty and $h_r = 0$. Without loss of generality, let the tool direction be in the $-z$ direction and the tooltip be positioned at the origin of $\mathcal{T}$ as indicated in figure 5. Further, we assume that T rotates around the z-axis of $\mathcal{T}$. Let s_{max} be the height of S_0, s_{min} the height of R as measured along the z-axis. Let $R_{XY} = \{(x, y) \mid \exists z : (x, y, z) \in R\}$ and let for all $(x, y) \in R_{XY}$ $R(x, y) = sup\{z \mid (x, y, z) \in R\}$. R can be obtained in $l = Round\_Up((s_{max} - (s_{min} + h_r))/h_c)$ stages. At each stage a slice is removed by following a plane parallel to the xy-plane or by following the boundary of $(F \oplus (-T)) \cup (R \oplus (-H))$. The slices are removed in a top down fashion such that in the first milling stage the tooltip is positioned at each point $(x, y) \in R_{XY}$ at height $\max(R(x, y), s_{max} - h_c - h_r)$. Hence, interference between H and S is avoided. Next, for $j = 2 \ldots l$ the tooltip is positioned in the jth milling stage at each point $(x, y) \in R_{XY}$ with $R(x, y) < s_{max} - (j - 1)h_c - h_r$ in the xy-plane at height $\max(R(x, y), s_{max} - jh_c - h_r)$. If the jth slice is being milled, the stock-in-progress S is always identical to R above the height $s_{max} - (j-1)h_c$ and upon completion of milling this slice, S is identical to R above $s_{max} - jh_c$. Since H can hit S only above that slice, interference between S and H is avoided.

Also the "slicing" strategy has been implemented without using Minkowski operations (see, e.g. [12]).

In the next section we define Minkowski operations on numerical functions and in section 5 we describe the implementation of both algorithms using numerical functions on $\mathbb{Z}^2$.

4 Minkowski operations on numerical functions

In gray-scale morphology [5, 13] the definitions of Minkowski operations on sets are modified to real-valued functions defined on finite subsets of $\mathbb{R}^n$, the Euclidean space of dimension n, and on $\mathbb{Z}^n$, the Euclidean grid of dimension n. These functions are called numerical functions. In the next section we use numerical functions on $\mathbb{Z}^2$ and Minkowski operations to describe milling algorithms.

Let $D_f \subset \mathbb{Z}^2$ denote the domain of the function f. Let the **translation** of f by a point (s, t) be denoted by $f_{s,t}$. $f_{s,t}$ is defined as

$$D_{f_{s,t}} = \{(x, y) \mid (x - s, y - t) \in D_f\}$$

and for all $(x, y) \in D_{f_{s,t}}$

$$f_{s,t}(x, y) = f(x - s, y - t).$$

Given a set S, the reflected set $\hat{S} = \{(x, y, -z) \mid (x, y, z) \in S\}$ is symmetrical to S with respect to the horizontal plane of coordinates.

The maximum and the minimum of a finite collection of numerical functions are defined as follows. Let f_k denote an element of a finite collection of numerical functions. Then the **maximum of a finite collection of numerical functions** is defined at (x, y) as $MAX(f_k)(x, y) = \max(f_k(x, y))$, if there exists at least one k such that $(x, y) \in D_{f_k}$ and where the maximum is over all such k. Otherwise $MAX(f_k)(x, y)$ is undefined at (x, y).

The **minimum of a finite collection of numerical functions** is defined at (x, y) as

$MIN(f_k)(x,y) = \min(f_k(x,y))$, if $(x,y) \in D_{f_k}$ for all k. Otherwise $MIN(f_k)(x,y)$ is undefined at (x,y).

The volume that is represented by a numerical function f on $\mathbb{Z}^2$ is called its **umbra** $U(f)$. This is the set of all points that lie below f, i.e.,

$$U(f) = \{(x,y,z) \mid (x,y) \in D_f \wedge z \leq f(x,y)\}.$$

In mathematical morphology the **Minkowski addition $f \oplus g$ of two numerical functions** f and g on $\mathbb{R}^2$ is defined as a dilation of f with g such that $U(f \oplus g) = U(f) \oplus U(g)$ as

$$D_{f \oplus g} = \bigcup_{(s,t) \in D_g} D_{f_{s,t}}$$

and for all $(x,y) \in D_{f \oplus g}$

$$(f \oplus g)(x,y) = \sup_{(s,t) \in D_g} \{f_{s,t}(x,y) + g(s,t)\}.$$

In mathematical morphology the **Minkowski subtraction $f \ominus g$ of two numerical functions** is defined as an erosion of f with g such that $U(f \ominus g) = U(f) \ominus \check{U}(g)$ as

$$D_{f \ominus g} = \bigcap_{(s,t) \in D_g} D_{f_{s,t}}$$

and for all $(x,y) \in D_{f \ominus g}$

$$(f \ominus g)(x,y) = \inf_{(s,t) \in D_g} \{f_{s,t}(x,y) - g(s,t)\}.$$

Next, we consider numerical functions on $\mathbb{Z}^2$. Given a grid size d, we associate numerical functions on $\mathbb{Z}^2$ with volumes by a mapping M from numerical functions on $\mathbb{Z}^2$ to numerical functions on $\mathbb{R}^2$. **The mapping M** is given by

$$M(f)(x,y) = f(Round\_Down(x/d), Round\_Down(y/d)),$$

where

$$D_{M(f)} = \{(x,y) \in \mathbb{R}^2 \mid \exists (i,j) \in D_f : id \leq x < (i+1)d \wedge jd \leq y < (j+1)d\}.$$

Let $\oplus_z$ and $\ominus_z$ denote the definitions of $\oplus$ and $\ominus$ for numerical functions on $\mathbb{Z}^2$ instead of numerical functions on $\mathbb{R}^2$ as given above.

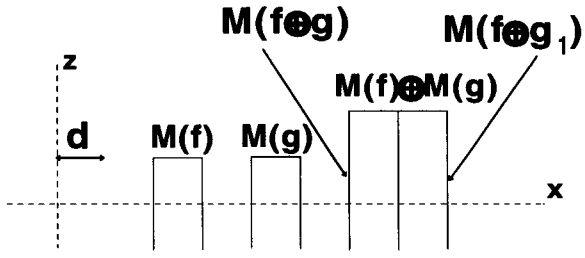

Figure 6: Let f and g denote numerical functions on $\mathbb{Z}$. $M(f) \oplus M(g) \neq M(f \oplus g)$ and $M(f) \ominus M(g) \neq M(f \ominus g)$. The umbra of $M(f \oplus g)$ is constructed by translating the umbra of M(f) by each element of the umbra of M(g) and then taking the union of all the resulting translates. Hence, $M(f) \oplus M(g) = M(MAX(f \oplus g, f \oplus g_1))$, where g_x denotes the translate of g by x, i.e., $g_x(s) = g(s-x)$. Since the umbra of $M(f \ominus g)$ is constructed by translating the umbra of M(f) by each element of the umbra of M(g) and then taking the intersection of all the resulting translates, $M(f) \ominus M(g) = M(MIN(f \ominus g, f \ominus g_1))$.

Note that for numerical functions on $\mathbb{Z}^2$ with finite domains, sup and inf are identical with max and min, respectively.

Figure 6 shows $M(f) \oplus M(g) \neq M(f \oplus_z g)$ and $M(f) \ominus M(g) \neq M(f \ominus_z g)$. In order to satisfy $M(f) \oplus M(g) = M(f \oplus_z g)$ and $M(f) \ominus M(g) = M(f \ominus_z g)$ we define Minkowski operations for numerical functions on $\mathbb{Z}^2$ as follows.

The **Minkowski addition $f \oplus g$ of two numerical functions** f **and** g **on** $\mathbb{Z}^2$ is defined as

$$f \oplus g = MAX(f \oplus_z g, f \oplus_z g_{1,0}, f \oplus_z g_{0,1}, f \oplus_z g_{1,1})$$

and the **Minkowski subtraction $f \ominus g$ of two numerical functions** f **and** g **on** $\mathbb{Z}^2$ is defined as

$$f \ominus g = MIN(f \ominus_z g, f \ominus_z g_{1,0}, f \ominus_z g_{0,1}, f \ominus_z g_{1,1}).$$

Given two numerical function f and g on $\mathbb{Z}^2$ with finite domains containing m and n elements, respectively, computing $f \oplus g$ requires $O(mn)$ time. Also computing $f \ominus g$ requires $O(mn)$ time.

5 Toolpath computation

In this section we discuss the implementation of the algorithms described in section 3. Given a fixed tool orientation, we will use numerical functions $f : D_f \rightarrow \mathbb{R}$, where $D_f \subset \mathbb{Z}^2$ is a finite set, d is a grid size to approximate the stock-in-progress S, the freeform shape geometry F, the toolholder H and the milling tool T. We also use numerical functions on $\mathbb{Z}^2$ to describe toolpath surfaces. There exist several methods [1, 6] to extract toolpaths from these numerical functions.

Since the obtained prototype should always enclose F, we represent the actual stock-in-progress S by a minimal numerical function s^e such that the umbra of $M(s^e)$ encloses S. Since collision between F and T should be avoided we also represent F and $-T$ by such enclosing numerical functions f^e and t^e. We can choose the maximal value of t^e to be equal to zero. Also collision between S and H should be avoided. Hence, we use a minimal enclosing numerical function h^e to represent $-H$. We assume that the maximal value of h^e is equal to $-(h_r + h_c)$ as indicated in figure 7, i.e., $h^e(x, y) = -(h_r + h_c)$ for all $(x, y) \in D_{t^e}$. Because s^e must be updated such that it encloses the actual stock-in-progress we need also a maximal numerical function t^i such that the umbra of $M(t^i)$ is enclosed by $-T$. Since T is rotation symmetric around the z-axis of T, $-t^i$ approximates T and $\hat{T} = -T$. Figure 7 illustrates these representations of T and H. Also a toolpath surface is represented by a numerical function p. The Minkowski addition and subtraction of these functions can be computed straightforwardly.

With these numerical functions the "remove as much material as possible" milling strategy can be described by the following algorithm that computes a set of toolpath surfaces p_i and the final result r of the milling process (recall, that $P_i = (F \oplus (-T)) \cup (S_i \oplus (-H))$ and that $S_{i+1} = P_i \ominus T$).

```
i:=0; s^e:="the initial stock";
(* Use t^e to compute F ⊕ (−T) *)
f^d := f^e ⊕ t^e;
REPEAT
   (* Use t^e to avoid collision *)
   p_{i+1} := MAX(f^d, s^e ⊕ h^e);
   (* Use t^i to update
   the stock-in-progress *)
   s^e_old := s^e; s^e := MIN(s^e, p_{i+1} ⊖ t^i);
   i := i + 1;
   (* Stop if no more material
   can be removed *)
UNTIL s^e = s^e_old;
(* s^e is the final result
of the milling process. *)
r := s^e;
```

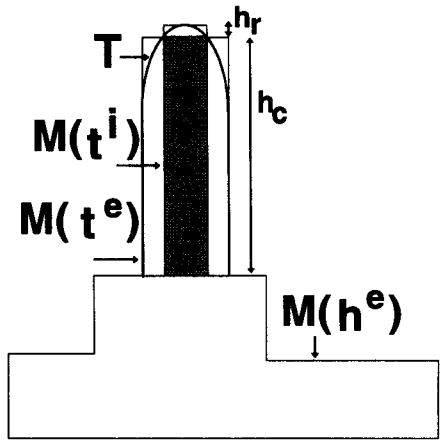

Figure 7: The milling tool T is represented by the minimal numerical function t^e such that the umbra of $M(t^e)$ encloses $-T$ and the maximal numerical function t^i such that the umbra of $M(t^i)$ is enclosed by $-T$. $-H$ is represented by the minimal numerical function h^e such that the umbra of $M(h^e)$ encloses $-H$. $-\min\{t^i(j) \mid j \in D_{t_i}\}$ approximates h_r and $\max\{h^e(j) \mid j \in D_{h^e}\} - \min\{t^i(j) \mid j \in D_{t_i}\}$ approximates h_c.

Below, we describe an algorithm that computes "sliced" toolpath surfaces. With these toolpath surfaces the same final result r is obtained. If a toolpath surface p_i is contained in a slice, the toolholder can hit the stock-in-progress only above that slice. Therefore, we compute the toolpath surfaces slice by slice. Let s_{max} denote the maximal height of the initial stock s^e_0, let s_{min} denote the minimal height of f^e. In this algorithm we use the following SLICE and SLI operations on numerical functions.

Let $SLICE^{to}_{from}(f)(x, y) = f(x, y)$, if $from \leq f(x, y) \leq to$. Otherwise let $SLICE^{to}_{from}(f)(x, y)$ be undefined.

Let $SLI^{to}_{from}(f)(x, y) = f(x, y)$, if $from \leq f(x, y) \leq to$. Let $SLI^{to}_{from}(f)(x, y) = from$, if $from > f(x, y)$. Otherwise let $SLI^{to}_{from}(f)(x, y)$ be undefined.

The algorithm proceeds as follows.

i:=0; s^e:="the initial stock";
$f^d := f^e \oplus t^e$;
(* l denotes the number of slices *)
$l := Round\_Up((s_{max} - (s_{min} + h_r))/h_c)$;
(* compute the boundaries of the first slice
and compute $F \oplus (-T)$ *)
$to := s_{max}$; $from := to - h_c$; $q := f^d$;
(* the top slice can be milled without
any toolholder collision check *)
(* extract the toolpath surface p_1
from $q = f^d$ *)
$p_1 := SLI^{to}_{from-h_r}(q)$;
(* update the stock-in-progress *)
$s^e := MIN(s^e, p_1 \ominus t^i)$;
FOR $i := 1$ TO $l - 1$
LOOP
  FOR ALL $(i,j) \in D_{s^e \oplus t^i}$
  LOOP
    $q(i,j)$:=
    $MAX(q, SLICE^{to}_{from}(s^e) \oplus h^e))(i,j)$;
  END LOOP;
  (* If we follow the toolpath surface q in
  the next slice, the toolholder
  does not collide with the
  stock-in-progress above that slice *)
  $to := from$; $from := to - h_c$;
  (* extract the toolpath surface p_{i+1}
  from q *)
  $p_{i+1} := SLI^{to-h_r}_{from-h_r}(q)$;
  (* update the stock-in-progress *)
  $s^e := MIN(s^e, p_{i+1} \ominus t^i)$;
END LOOP;
$r := s^e$;

We have checked the "remove as much material as possible" and the "slicing" algorithm for the 2D case by an implementation using numerical functions on $\mathbb{Z}$. Figure 8 shows an example of a toolpath surface q which has been computed with the "slicing" algorithm.

If the initial stock is not a block, positioning the tool tip at some grid elements (k,l) may be of no use. Therefore, each path surface $p_i(k,l)$ should be marked using a Boolean function indicating whether the tool intersects the stock-in-progress. The actual path will be extracted from this marked toolpath surface.

The computing time of this algorithm can be shown to be $O(MN)$ where M is the number of stock grid elements and N is the number of holder grid elements.

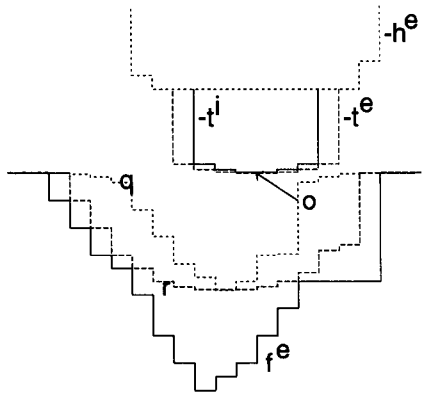

Figure 8: Illustration of the algorithm in 2D. A toolpath "surface" q and the final result r computed by the slicing algorithm. If o is positioned on q interference between the freeform shape F and the milling tool T, as well interference between the toolholder H and the stock-in-progress S is avoided.

We expect that for our Sculpturing Robot system the machining time with the paths that are generated by the "slicing" strategy will be less than the machining time with the paths that are generated by the "remove as much material as possible" strategy. This can be explained as follows. The paths consist of a number of straight line segments. Our Sculpturing Robot system reduces the speed of the tool to zero at the end of each straight line segment. Since the paths that are generated by the "slicing" strategy contain far less line segments, we expect that its machining time will be shorter. We will implement both milling strategies as part of a new motion planner for the Sculpturing Robot system and compare the machining time and the computing time of both strategies.

6 Efficient toolpath computation using Z-pyramids

The most CPU-intensive part of the "slicing" algorithm of the previous section is the computation of q by the inner loop

FOR ALL $(i,j) \in D_{s^e \oplus t^i}$
LOOP
  $q(i,j) := MAX(q, s \oplus h^e))(i,j)$;
END LOOP;

where $s = SLICE^{to}_{from}(s^e)$. This computation can be optimized using a Z-pyramid representation [9] for q, s and h^e. The basic idea of the Z-pyramid is to use the original numerical function as finest level of the pyramid and then combine four z values at each level into one z value at the next coarser level by choosing the maximal z value. At the coarsest level of the pyramid there is one single z value which is the maximal value of the numerical function. Let a minimal Z-pyramid be defined by choosing at the next coarser level the minimal Z-value instead of the maximal Z-value.

In order to facilitate to read the description of a Z-pyramid below we define its numerical functions on the set of integer rectangles $\{[x1, x2] \times [y1, y2] \mid x1, x2, y1, y2 \in \mathbb{Z} \wedge x1 \leq x2 \wedge y1 \leq y2\}$.

Definition of maximal Z-pyramid

Let, for a numerical function f, its maximal Z-pyramid be given by k numerical functions f_i, $i = 1 \ldots k$, where f_1 is a numerical function having one single value which is the maximal value of f and f_k corresponds with f. The Z-pyramid is derived from the original function as follows. Let $V(t, i)$ denote the integer interval starting at t that contains 2^{k-i} elements, i.e.,
$V(t, i) = [t, t + 2^{k-i} - 1]$.

Let f_k be defined as: $f_k([x, x] \times [y, y]) = f(x, y)$ for all $(x, y) \in D_f$. Let f_i, $i = 1 \ldots k - 1$ be defined as $f_i(V(2^{k-i}x, i) \times V(2^{k-i}y, i)) = \max(f_{i+1}(V(2^{k-i}x, i+1) \times V(2^{k-i}y, i+1)$, $f_{i+1}(V(2^{k-i}x + 2^{k-(i+1)}, i+1) \times V(2^{k-i}y, i+1)$, $f_{i+1}(V(2^{k-i}x, i+1) \times V(2^{k-i}y + 2^{k-(i+1)}, i+1)$, $f_{i+1}(V(2^{k-i}x + 2^{k-(i+1)}, i+1) \times V(2^{k-i}y + 2^{k-(i+1)}, i+1))$

End of definition

Minimal Z-pyramids can be defined analogously. Let $<$ denote the usual partial ordering relation for numerical functions on $\mathbb{Z}^2$. Hence, for two numerical functions f_{i-1} and f_i in a maximal Z-pyramid holds $f_i \leq f_{i-1}$ and for two numerical functions f_{i-1} and f_i in a minimal Z-pyramid holds $f_i \geq f_{i-1}$.

Let a Z-pyramid representation for s and h^e be given. We assume that all these Z-pyramids have the same level (this can be achieved by adding levels with the same single Z value). From the Z-pyramid definition it follows that $s_i \leq s_{i-1}$ and $h^e_i \leq h^e_{i-1}$. From the definition of Minkowski addition follows $s_i \oplus h^e_i \leq s_{i-1} \oplus h^e_{i-1}$. Let a minimal Z-pyramid be given for q with the same level as the Z-pyramids for s and h_e. From the definition of minimal Z-pyramid follows $q_i \geq q_{i-1}$. Hence, q and its minimal Z-pyramid can be computed by starting the following recursive algorithm at level 1 with appropriate values for $x1$, $x2$, $y1$ and $y2$.

```
PROCEDURE UPDATE_Q(x1,x2,y1,y2,l)
(* l denotes the Z-pyramid level *)
IS
BEGIN
w := s_l ⊕ h^e_l([x1, x2] × [y1, y2]);
IF w < q_l([x1, x2] × [y1, y2]) ∧ l < k
THEN
   (* updating q and its Z-pyramid
   is not necessary *)
   NULL;
ELSE
   (* update q_l and refine
   the computation of q *)
   q_l([x1, x2] × [y1, y2]) := w;
   (* Let "/" denote integer division,
   e.g. 5/2=2 *)
   UPDATE_Q
   (x1,(x1+x2)/2,y1,(y1+y2)/2,l+1);
   UPDATE_Q
   ((x1+x2)/2+1,x2,y1,(y1+y2)/2,l+1);
   UPDATE_Q
   (x1,(x1+x2)/2,(y1+y2)/2+1,y2,l+1);
   UPDATE_Q
   ((x1+x2)/2+1,x2,(y1+y2)/2+1,y2,l+1);
END IF;
END UPDATE_Q;
```

This Z-pyramid algorithm has been tested for the 2D case with our implementation of numerical functions on $\mathbb{Z}$.

7 Conclusions and further research

We have shown that Minkowski operations form a powerful tool to specify the freeform shape machining problem and to describe and compare freeform shape machining algorithms. We have described the well-know "remove as much material as possible" machining strategy and the well-know "slicing" machining strategy using the Minkowski addition and subtraction. Since the latter strategy generates toolpaths, that can be followed by the tool in less time, we have described an efficient implementation for the Minkowski addition using Z-pyramids.

Further current research includes the selection of promising tool access directions [3, 15, 17], and the extraction of toolpaths from the computed toolpath surfaces [1, 6]. The "remove as much material as possible" strategy and the "slicing" strategy will be implemented using Z-pyramids as part of a new motion planner for the Sculpturing Robot system. The machining time and the computing time of both strategies will be compared.

References

[1] J.J. Cox, Y. Takezaki, H.R.P. Ferguson, K.E. Kohkonen, and E.L. Mulkay. Space-filling curves in tool-path applications. *Computer-Aided Design*, 26(3):215–224, 1994.

[2] H. Edelsbrunner and E.P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[3] J.G. Gan. Set-up orientations of workpieces for machining by three-axis numerical control machines. *Journal of Design and Manufacturing*, 2:59–69, 1992.

[4] P.K. Ghosh. A solution of polygon containment, spatial planning, and other related problems using Minkowski operations. *Computer Vision, Graphics and Image Processing*, 49:1–35, 1990.

[5] R.G. Giardina and R.D. Dougherty. *Morphological Methods in Image and Signal Processing*. Prentice Hall, 1988.

[6] J.G. Griffiths. Toolpath based on Hilbert's curve. *Computer-Aided Design*, 26(11):839–844, 1994.

[7] A. Kaul. Minkowski sums: A simulation tool for CAD/CAM. In G.A. Gabriele, editor, *Computers in Engineering - Volume 1*, pages 447–456, New York, 1992. ASME.

[8] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[9] D. Laur and P. Hanrahan. Hierarchical splatting: a progressive refinement algorithm for volume rendering. *Computer Graphics*, 25(4):285–288, 1991. (SIGGRAPH '91).

[10] T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transaction on Computers*, C-32(2):108–120, 1983.

[11] J. Menon, R.J. Marisa, and J. Zagajac. More powerful solid modeling through ray representations. *IEEE Computer Graphics and Applications*, 14(3):22–35, 1994.

[12] T. Saito and T. Takahashi. NC machining with G-buffer method. *Computer Graphics*, 25(4):207–216, 1991. (SIGGRAPH '91).

[13] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.

[14] J.W.H. Tangelder and J.S.M. Vergeest. Robust NC path generation for rapid shape prototyping. *Journal of Design and Manufacturing*, 4:281–292, 1994.

[15] J.W.H. Tangelder, J.S.M. Vergeest, and M.H. Overmars. Computation of voxel maps containing tool access directions for machining free-form shapes. In *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, New York, 1996. ASME.

[16] A.M Vepsäläinen. An application of morphological filters to NC-programming. In P.D. Gader, editor, *SPIE Vol. 1350 Image Algebra and Morphological Image Processing*, pages 177–183, Washington, 1990. SPIE - The international Society for Optical Engineering.

[17] T.C. Woo. Visibility maps and spherical algorithms. *Computer Aided Design*, 26(1):6–16, 1994.

[18] C.F. You and C.H. Chu. An automatic path generation method of NC rough cut machining from solid models. *Computers in Industry*, 26:161–173, 1995.