

**Surface-adaptive and  
Collision-avoiding Path Planning  
for Five-axis Milling**

Peter Bollweg,  
Dino Hasanbegovic,  
Heinrich Müller, Mattias Stöneberg

Forschungsbericht Nr. 808/2006  
January 2006



# **Surface-adaptive and Collision-avoiding Path Planning for Five-axis Milling**

Peter Bollweg, Dino Hasanbegovic, Heinrich Müller, Mattias Stöoneberg

Universität Dortmund

FB Informatik LS VII

D-44221 Dortmund

Germany

Tel.: +49 - 231 - 755 6134

Fax.: +49 - 231 - 755 6321

e-Mail: [mueller@ls7.informatik.uni-dortmund.de](mailto:mueller@ls7.informatik.uni-dortmund.de)

Forschungsbericht Nr. 808/2006

January 2006

## **Abstract**

Automatic path planning for 5-axis-milling is a difficult problem if it is tried to use all degrees of freedom in an optimal way. Existing computer-based planning still require a considerable amount of interaction by experienced users, and use quite adhoc heuristics useful in practice in the automatic part. This paper presents two approaches to automatic path planning with emphasis on complex free-formed workpieces: a graph path covering approach and an iso-potential lines approach. They are based on the modelling of path-planning as optimization problems which include the behavior of the tool and the production machine with respect to kinematic aspects and collision avoidance. The algorithmic implementation of the approaches is based on discretization of the workpiece surface by a triangular mesh. A particular advantage of the iso-potential-curve approach is the robust calculation due to the used implicit representation, and its lower computing times. The graph path covering approach is less restrictive with respect to the set of feasible solutions in which an optimum one is searched.

# 1 Introduction

Milling is a process of production technology. A milling tool is moved through a stock of material in a way that a desired workpiece is generated by removing material from the stock. The tool is moved by a milling machine which is computer-controlled. The task of automatic path planning is to find a suitable path of the tool for that purpose algorithmically.

In practice, 3-axis milling is well introduced. 3-axis milling restricts the tool movement (or, equivalently, the workpiece movement) to left/right, forward/backward, and up/down. 5-axis milling additionally allows the rotation of the tool (or, equivalently, of the workpiece).

Automatic path planning for 5-axis-milling is a difficult problem if it is tried to use all degrees of freedom in an optimal way. Existing computer-based planning systems like AutoCAD 2006, Openmind Hypermill 9.0 of HyperCAD 9.0, the integrated CAM module of CATIA V5 R14, Solid CAM by Solidworks 2006, Pro/NC in Pro/ENGINEER, and SolidEdge V.18, NX CAM ISV by UGS Unigraphics Solutions still require a considerable amount of interaction by experienced users, and apply quite adhoc heuristics useful in practice in the automatic part, neglecting the principal modeling and algorithmic analysis of the problem.

This paper focuses on the final stage of milling close to the workpiece surface. It presents two approaches to automatic path planning with emphasis on complex free-formed workpieces and worst-case configurations in this sense: the *graph path covering approach* and the *iso-potential lines approach*. They are based on the modeling of path-planning as optimization problems which include the behavior of the tool and the production machine with respect to kinematic aspects and collision avoidance. A general view of this kind has been unusual until recently. The two approaches can be distinguished in the type of paths: roundtrips on a mesh and iso-potential curves on a mesh. An important difference between both approaches is that the graph path covering approach is less restrictive with respect to the set of feasible solutions in which an optimum one is searched. The iso-potential lines approach controls the set of feasible paths by heuristic parameters which prefers a certain shape of paths. If the parameters are used in an intuitive way, the search space is reduced, but the calculation times are considerably less than those of the graph path covering approach. The main difference of both approaches to existing ones is that not just the covering property, but further properties, like collision avoidance and configuration of the

tool and the production machine are taken into account. A particular advantage of the iso-potential-curve approach is the robust calculation due to the implicit representation used.

Section 2 gives a survey on related work. Section 3 is devoted to a general model of automatic 5-axis path planning. Section 4 presents basic subproblems and corresponding solutions which are used by both subsequent approaches of solution of the path planning problem. The first of those solutions, construction of a milling path based on a Hamiltonian surface path with constraints, is subject of section 5. Section 6 is devoted to second solution, based on isolines of potential functions which model constraints. Section 7 concludes the paper with remarks on possible future work.

## 2 Related Work

Path planning is a central task of computer-aided milling, and considerable work has been performed in the past, cf. e.g. the surveys by Marshall/Griffiths [Mar94], Dragomatz/Mann [Dra97], and the book by Choi/Jerard [Cho99]. Nevertheless, the problem cannot be considered as solved yet, in particular for complex free-formed shapes which require manipulation with five axes of freedom.

The task of path planning is finding a motion path of a tool which yields a good approximation of a desired workpiece if applied to an initially given workpiece. A tool path consists of a continuous sequence of tool configurations. A tool configuration is given by a location and orientation of the tool in space. The tool in motion along the tool path has to transform the workpiece from its given shape into a desired shape. A tool path is suitable to perform this transformation if the following constraints are fulfilled:

1. To every point  $\mathbf{p}$  on the desired shape a point  $\mathbf{q}$  exists on the cutting part of the tool so that the distance between  $\mathbf{p}$  and  $\mathbf{q}$  is less than a given bound  $\varepsilon > 0$ .
2. The tool never penetrates the desired workpiece with its cutting part.
3. The tool never collides with the current workpiece outside the cutting part.

The cutting part of the tool is the region of the tool which can erase material if it is in contact with the workpiece, in contrast to the rest of the tool which should not get in contact with the workpiece. Usually the top of a milling cutter is used as cutting part whereas the shank of the milling tool is not. The bound  $\varepsilon > 0$  defines the precision achieved by a path of this type.

Two main situations can be distinguished: path planning *close to the desired workpiece surface* and planning of paths *far from the desired workpiece surface*. The *close-to-surface situation* means that the superfluous material can be erased by a path along the desired surface, that is, the layer of material to be removed is thin enough so that the tool can always be in contact with the desired surface. In the *far-from-surface situation*, the difference between the current and the desired workpiece is so thick that the tool generally cannot reach the desired surface. Since discrete displacement fields focus on the boundary of a surface, our interest lies on the close-to-surface case. Most contributions to the field concern the close-to-surface case which is not surprising since it is of higher importance for the final quality of the produced surface than the far-from-surface process. The aspect of far-from-surface removal of material has been addressed for example by Tangelder et al. [Tan98] and Flutter/Todd [Flu02].

Comprehensive approaches to modeling of the path planning problem are rare. A recent exception is the model by Kim et al. [Kim03]. The typical approaches to close-to-surface path planning are characterized by *global reduction* of the space of all paths by heuristic constraints, and local path selection, within the constraints, by *local optimization*. Typical constraints concern the selection of the tool [Jen02, Gla99], the milling approach and the milling strategy [Hos92, Hel91]. One approach is to subdivide a given surface into processing objects, each of them satisfying a special set of constraints [Sto99]. Path planning is then performed on each processing object in separate, with a strategy suitable to the constraints. The approach of processing objects is particularly useful if standard processing objects can be identified which can be pre-processed and re-used [Mey99]. The possibility of re-use, however, is more likely on regular shapes than on free-formed shapes. Another aspect is adaptation of the desired shape to the requirements of milling [Elb97], in particular if the shape cannot be produced as it is.

*Local optimization* can have several goals: local tool fitting, avoidance of collisions, choice of distance between neighboring segments, minimization of idle path length, and compensation of tool deformation.

The goal of *local tool fitting* is to approximate the workpiece surface well by the cutting part of the tool, under avoidance of penetration of the workpiece. Approaches to reaching the goal have been tilting of the tool along a given path [Lee97, Li94, Yan99], and a priori avoidance of the problem by a rolling ball approach [Gra02] or by analytical exclusion [Pot99]. There is also some relation to tasks of assembling, like for instance drilling in a screw, where motion paths can be generated by online collision detection and response [Len99].

Solutions proposed for the problem of *suitable distances between neighboring path segments* have been the choice by error analysis [Lee98], surface-adaptive determination of the distance of iso-planar path segments using isophotes [Din03], and a vector field approach [Kim02, Kim03].

By *idle motion* we mean those parts of the overall motion of the tool where the tool is not in contact with the workpiece. These parts, which connect segments of the milling path, have to be minimized. This task seems to be related to the NP-hard traveling salesman problem [Gar79]. A recent treatment can be found in [Par02].

Usually path planning assumes rigid tools. However, in practice milling or grinding tools are deformed during the process. An approach to taking into consideration *tool deformation* is to perform path planning with a rigid tool. With the resulting path a milling or grinding simulation is carried out which takes into consideration deformation based on the calculation of acting forces and tool reaction. Approaches to modeling of tool engagement conditions can for example be found in [Wei01] for the case of milling simulation. The result of simulation can be used for correction of the tool path. Further simulation and possibly iteration of this approach can be used in order to verify and improve the result. However, procedures like this can be time-consuming so that they are not in widespread use up to now.

*Collision avoidance* concerns the tool outside the cutting part and other parts of the milling machine which must not get in touch with the workpiece. Two types of approaches can be distinguished: *offline planning* and *online planning* of collision avoidance.

Methods of *offline collision planning* explicitly pre-calculate a representation of the free-space in order to use it for finding a suitable path [Lat91]. *Online planning* means that, during calculation of a path, tests for intersection with the obstacle are performed and, dependent on the result, the next step on the path is chosen collision-free. For quick intersection tests and dis-

tance calculations, algorithms and data structures based on bounding volumes like spheres, axes-parallel bounding boxes, and oriented bounding boxes are applied [Len99]. Also the computing power of graphics hardware available for e.g. z-buffering is sometimes used. Online collision avoidance has been used by several authors for path planning of milling processes by simulation [Ho01, Elb94, Lau02, Li94]. Other suggestions use potential fields [Chi02] or interpret collision avoidance as a visibility problem [Elb94].

Our work integrates many of the aspects mentioned. It focuses on finding paths for close-to-surface milling. It assumes that the tool size and a region of interest on the workpiece have been already selected. With respect to collision avoidance it has some similarity with Balasubramaniam et al. [Bal02]. In its generality it resembles the approach of Kim et al. [Kim03]. They use a vector field view approach, but do not use potential fields which are more stable and flexible.

### **3 Modeling of the Problem of 5-Axis Path Planning**

This chapter presents a general model of path planning. For that purpose, the workpiece (section 3.1) and the milling tool (section 3.2) are described in mathematical representations. The representations intend to specify the shapes of the workpiece and the milling tool precisely. Later-on, the precise representations will be partly replaced with discrete approximations which allow to apply methods of combinatorial geometry for finding solutions of the path planning problem. The next step is modeling the contact between the workpiece surface and the tool surface which leads to the definition of contact points and accessibility (section 3.3). Then the production machine is included using the concept of configuration space (section 3.4). Based on this, several possibilities of representing tool movement paths are presented (section 3.5). After introducing concepts specifying the production error (section 3.6), a definition of path planning problem can be given (section 3.7).

### 3.1 The Workpiece

A *workpiece* is given as a volume  $V$  with a closed, orientable and continuous surface  $S$  with feature lines (figure 1).  $S$  is smooth at every point except possibly for the feature lines. *Smooth* at a point means that the surface is locally  $C^k$ ,  $k \geq 1$ , at the point, that is the  $k$  first derivatives exist.

A closed surface is *orientable* if and only if subdivides the space into two disjoint parts, the interior and the exterior of the surface.

$S$  is called *locally  $C^k$*  at a point  $\mathbf{p} \in S$  if a local parametrization  $S_{\mathbf{p}}(u, v)$  of an environment of  $\mathbf{p}$  on  $S$  exists which is  $C^k$ .

The region  $S^I \subseteq S$  which is subject of path planning is called *region of interest*. The rest of  $S$  has to be taken into account, too, in order to avoid collisions. The reason for introducing the region of interest is that it often happens that  $S$  is not completely accessible for the tool in just one position relative to the production machine.

A point on the surface is *concave* if the tangent planes at the point do not traverse the exterior in a small environment. It is *convex* if the same holds for the interior. If every tangent plane intersects the environment as well as the workpiece volume, then the point is called *heterogeneous*. If a heterogeneous point is locally  $C^2$ , it is a *saddle point*.

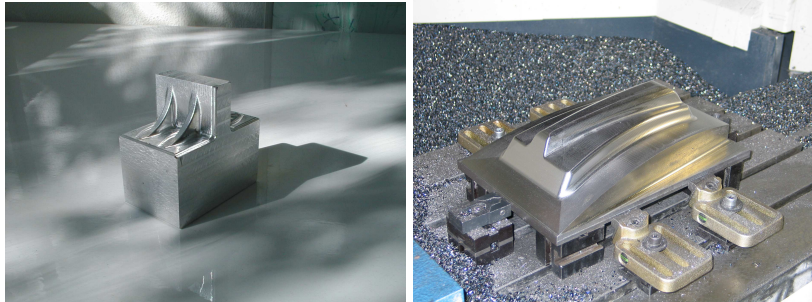


Figure 1: Workpieces with sharp concave and convex edges.

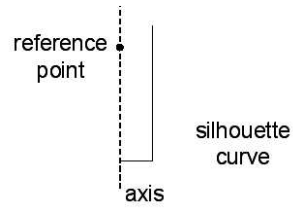


Figure 2: Definition of a tool.

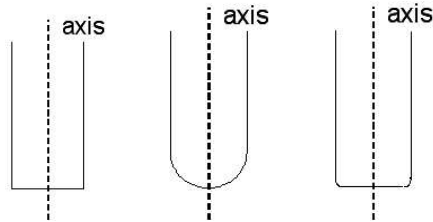


Figure 3: Tool shapes: ball end cutter, torus cutter, flat end cutter.

### 3.2 The Tool

A *tool* is represented by a closed rotational surface  $T$  given by an axis segment  $A$  and a silhouette curve  $s$  (figure 2). Typical tools are *ball end cutters*, *torus cutters*, and *flat end cutters* (cylindrical cutters) (figure 3). The tool has a *cutting region* where it is allowed to be in contact with the workpiece. A tool has a *reference point*  $r_T$  which is located on its axis. The *tip* of the tool is the intersection point of the tool axis with the tool surface at the cutting end.

For tool-related collision detection, the tool may be extended by a tool holder so that tool and tool holder together form a rigid body.

### 3.3 Contact Points and Local Accessibility

A tool and the workpiece surface are in contact at some point, a *contact point*, if they share the point and do not penetrate each other within a contact region on

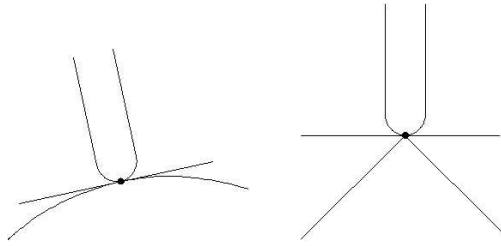


Figure 4: A contact point between a tool and a workpiece surface.

the tool (figure 4). The *contact region* is a subregion of the cutting region of the tool around the contact point. It will be characterized more precisely later-on. If a surface point is a contact point, it is called *locally accessible*.

A necessary condition for a contact point is that the workpiece surface and the tool surface share a tangent plane at the point. If both are  $C^1$  at the point, the tool contact configuration is uniquely described by the pair of points plus a rotational angle of the tool around the normal of the workpiece surface, since tangent planes are unique in that case. Otherwise two additional degrees of freedom exist. This case happens e.g. for flat-end cutters or at convex ridges of the surface.

A point on a surface is called *locally non-accessible* if the tool cannot be in contact with the surface at the point. The border between locally accessible and locally non-accessible points is called *accessibility border*.

### 3.4 Production Machine and Configuration Space

A *production machine*  $M$  controls the movement of the tool by a program which provides the machine with a sequence of *control parameter values*  $\mathbf{c}_i = (c_{i,1}, \dots, c_{i,m})$ ,  $i = 1, \dots, n$ , which specify a corresponding *configuration* of the machine. From those data values, a continuous  $m$ -dimensional control function over time is interpolated by the machine control unit, which induces a continuous motion of the machine. The time steps  $t_i$  which have to be assigned to every  $\mathbf{c}_i$  are material-dependent, and are not subject of the stage of path planning which is subject of this paper. They can e.g. be found by sim-

ulation based on the result of our path planning approaches.

The *position of a tool* is defined by a translation  $\mathbf{t}$  and rotation  $\mathbf{R}$  which specify the location relative to an initial location in a world coordinate system.

The production machine implements *direct kinematics functions*

$$\mathbf{t} = \mathbf{t}(\mathbf{c}), \mathbf{R} = \mathbf{R}(\mathbf{c}).$$

The *feasible configuration space*  $C(M)$  of the production machine  $M$  is the set of all parameters  $\mathbf{c}$  which induce feasible configurations of the production machine.

The *freespace*  $C^-(S, T)$  is the set of all parameters  $\mathbf{c}$  in  $C(M)$  which do not induce a collision between any involved objects.

The *freespace*  $C^-(S, T, \mathbf{p})$  of a *workpiece point*  $\mathbf{p} \in S$  is the set of all configurations  $\mathbf{c} \in C^-(S, T)$  for which  $(\mathbf{t}(\mathbf{c}), \mathbf{R}(\mathbf{c}))$  is a tool position in contact with  $\mathbf{p}$ . The tool configurations in  $C^-(S, T, \mathbf{p})$  are called *feasible*.

The *freespace*  $C^-(S, T, M)$  of a set  $M \subseteq S$  is the union of the freespaces of all points of  $M$ .

The minimal connected subsets of a freespace are called *freespace components*.

A point  $\mathbf{p} \in S$  is called *collision-safe* if in a sufficiently small environment  $U(\mathbf{p}) \subseteq S$  every freespace component of  $C^-(S, T, U(\mathbf{p}))$  is global on  $U(\mathbf{p})$ . A freespace component is *global* on  $U(\mathbf{p})$  if it contains a freespace configuration for every point  $\mathbf{q} \in U(\mathbf{p})$ . A point  $\mathbf{p} \in S$  is *collision-critical* if it is not collision-safe.

Analogously to a configuration space of the production machine, the *tool configuration space* is the set of all parameters  $(\mathbf{t}, \mathbf{R})$ . The freespace terminology is transferred in a canonical way to the tool configuration space.

A configuration of the tool or the production machine is called *globally accessible* if it can be reached from a basic configuration of rest.

### 3.5 Tool Movement Paths

A tool movement path describes the movement of the tool in space. The movement of the tool can be described in different ways (section 3.5.1). A particular

aspect of motion planning is the "fairness" of the resulting path which is addressed in section 3.5.2.

### 3.5.1 Representations of Tool Movement Paths

The different representations of the tool movement paths result from different combinations of geometric parameters.

#### Representation in the tool movement space

A *tool movement path* (TMP) in the tool movement space is a continuous curve  $\mathbf{P}$  in the tool movement space  $(\mathbf{t}, \mathbf{R})$ .

The TMP implies a *reference point curve*  $\mathbf{r}_T$ . A parametrization of the TMP by the length of the reference point curve is called a *geometric parametrization of the TMP*.

The *geometric translational acceleration* is the second derivative of  $\mathbf{r}_T$  in curve-length parametrization. Its absolute value is the geometric curvature of  $\mathbf{r}_T$  [Bla73, Car76].

The *geometric rotational acceleration* is the difference between the second derivatives of the reference point curve and of the curve of the tip of the tool.

#### Contact-based representation

The *contact-based representation* of a TMP for  $C^1$ -surfaces and  $C^1$ -tools is given by

$$(\mathbf{k}_S, \mathbf{k}_T, \phi_T)$$

where

$\mathbf{k}_S$  is a curve on the workpiece  $S$ , the *workpiece contact curve*,

$\mathbf{k}_T$  is a curve on the tool  $T$  which can be restricted to be a subset of a silhouette curve, and

$\phi_T$  is a function of rotational angles around the workpiece normal, all parametrized over the same domain.

For a parameter value  $t$ ,  $(\mathbf{k}_S(t), \mathbf{k}_T(t), \phi_T)$  specifies a contact where  $\mathbf{k}_S(t)$  and  $\mathbf{k}_T(t)$  are the same point.

A TMP can only be represented in a contact-based way if all its tool positions have a contact point. This does not hold if the tool has to be lifted off in order to reach a new path segment if a covering of the workpiece surface is not possible by one path.

The parameter domains of  $\mathbf{k}_S$  and  $\mathbf{k}_T$ , respectively, may have intervals mapped to the same curve point. This happens for instance if the contact point on the tool is the same for some time, or if the tool rotates while staying in contact with the same workpiece point.

$\mathbf{k}_S$  corresponds to the translational behavior of the TMP, while  $\mathbf{k}_T$  describes the rotational behavior of the TMP.

### Reference-point-based representation

The *reference-point-based representation* of a TMP is given by

$$(\mathbf{k}_S, \mathbf{k}_R)$$

where

$\mathbf{k}_S$  is a curve on the workpiece  $S$ , *the workpiece contact curve*,

$\mathbf{k}_R$  is a curve of the reference point of the tool  $T$ .

$\mathbf{k}_S$  and  $\mathbf{k}_R$  are dependent on each other.

### 3.5.2 Geometric Fairness of a TMP

The *fairness of a TMP* describes the variation of the acceleration of the tool or the moving parts of the production machine along the path. It is expressed mathematically by the acceleration of points on the cutting tool, like the tool reference point and the tool tip, or on moving parts of the machine. The dynamics of a point moving along a curve is described by derivatives of the curve in parameter representation where the parameter represents the time. The *speed* of a point on a curve  $\mathbf{k}(t)$  is given by the first derivative  $\mathbf{k}'(t)$ , and its *acceleration* is the second  $\mathbf{k}''(t)$ .

The time parameter can be chosen arbitrarily within the constraints of the production machine. In order to become independent from the parametrization, we take the curve length parametrization as reference. In the case of curve length parametrization, the speed vector has always length 1. The acceleration is the curvature vector, and its length is the curvature of the curve [Bla73, Car76].

The curvature vector of a curve on a surface can be split into a component colinear to the surface normal, and a component orthogonal to the curve tangent and the surface normal. The first one is called the *normal curvature*, the second one the *geodesic curvature*.

### 3.6 Production Error

Let us given an initial workpiece  $V^+$ , a desired workpiece  $V \subseteq V^+$ , and a family  $\mathcal{T}$  of feasible tool positions, for example those on a tool path. The *error volume* is defined by

$$V^e := V^+ - V - \bigcup_{T \in \mathcal{T}} T.$$

The *restricted error volume*  $V^{e,I}$  is the restriction of  $S^e$  to region of interest  $S^I$ , for instance by intersection with the union over all tool positions with contact points in  $S^I$ .

*Error measures* characterize the error volume [Jer89, Oli90, Hua94]. One possibility is considering the *one-sided distance error* which is the distance of the points of the produced surface from the desired surface. Another possibility is the *Hausdorff error* which is measured by the symmetric Hausdorff distance between the produced surface and the desired surface, which is more tight. A third possibility is the *displacement error*. It uses a displacement field on  $S$  which is intersection-free in  $V^e$ . A displacement field is given by a vector field on the surface [Ayasse02]. Typically, the vector assigned to a surface point is its normal, or at least a vector close to the normal if the normal is not possible because of penetrations. The error considers the lengths of the displacement vectors between  $S$  and the produced surface.

An observation is that at concave locations a toroidal tool seems well-suited, while at convex and saddle locations ball-shaped tools could be more favorable.

### 3.7 Path Planning

The problem of path planning can now be specified as follows.

#### Path Planning

**Input.** A given workpiece surface  $S^+$ , a desired workpiece surface  $S$ , a tool, a production machine, an error bound.

**Output.** A tool path  $P$  on  $S$  so that:

1. The tool intersects  $S^+$  only on its cutting region.
2. The tool path should not intersect itself.
3. The error between the produced surface  $S^0$  and  $S$  is less than the error bound.
4.  $P$  can be executed with a feasible movement of the production machine.
5.  $P$  takes into account the following objective functions:
  - (a) The geometric acceleration of the tool should be a fair function.
  - (b) The movements of the production machine should be a fair function.

Among the constraints on the path, some are mandatory or *hard*, and some are desirable or *soft*. *Hard constraints* are

- the accessibility borders,
- the boundary of a region of interest (which can possibly be adapted),
- collision freeness and feasibility of the production machine.

## 4 Subroutines

This chapter compiles algorithmic solutions of some components of the path planning problem which will be used as subroutines in the two overall solutions presented later. The subproblems are tool-workpiece accessibility detection at

a point  $\mathbf{p} \in S^I$  (section 4.1), tool-workpiece collision detection (section 4.2), restriction of the configuration space (section 4.3), collision analysis on curves (section 4.3.2), and extension of a workpiece contact curve to a tool path (section 4.4).

The definition of problem and concepts are based on smooth surface and path terminology. In addition, discrete solutions based on graphs are presented which will be part of the discrete implementation of the two approaches of solution. Both approaches use an approximation of the workpiece surface by a triangular mesh. Curves on the surface become polygonal curves on the mesh or even paths in the graph defined by the vertices and edges of the mesh.

In order to let readers not familiar with those concepts understand the following, here is a brief summary of the most important definitions. A *graph*  $G = (V, E)$  consists of a finite set  $V$  of *vertices* and a finite set  $E$  of *edges* which connect pairs of edges. In a geometric representation of a graph, the vertices may be points, and the edges may be straight lines connecting the points. A *path* in a graph is a sequence of vertices where every consecutive pair of points is connected by an edge. A *triangular mesh*, or briefly *mesh* in the following, is a surface composed of a finite number of triangular cells. The triangular cells are bounded by edges and vertices. The vertices and edges of a mesh define a graph. A *polygonal path* is given by a finite sequence of points, also called vertices, where every consecutive pair of points is connected by a line segment. Evidently, paths in a mesh graph are examples of polygonal chains.

## 4.1 Tool-workpiece accessibility detection

Local Accessibility detection at a point  $\mathbf{p} \in S^I$  has two versions: *existence* and *enumeration*.

The first version is the question for the existence of a tool configuration in which  $\mathbf{p}$  is a contact point.

The second version is the question for the local accessibility space of the  $\mathbf{p}$ . The *local accessibility space* of a point on  $S^I$  is the region of the tool-configuration-space which contains those tool configurations in which the point is a contact point.

The existence problem can be considered as a special case of the local accessibility space problem. One approach to its solution is to take one of the algo-

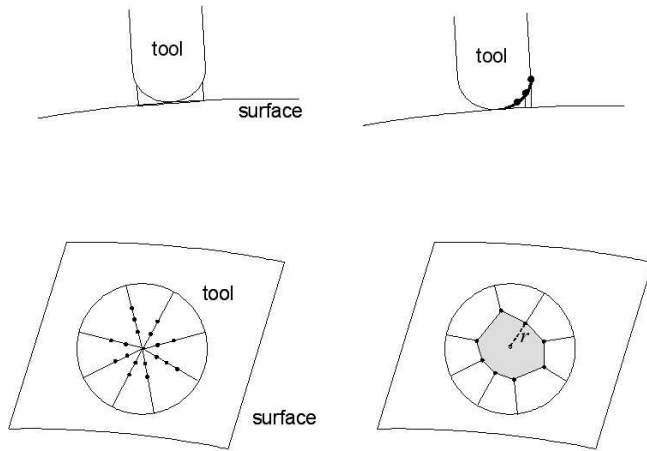


Figure 5: Accessibility calculation: Definition of a contact candidate region on the tool (top left), projection of sample points on the tool onto the workpiece surface (top right in side view, bottom left in top view), and the approximation of the error tolerance radius (bottom right).

rithms for the second problem, outlined in the following, but stopping search if a feasible configuration is found.

The enumeration problem can be expressed more precisely as follows.

**Accessibility space problem.**

**Input.** A point  $\mathbf{p}$  on the workpiece, a region  $R(T)$  of contact candidate points  $\mathbf{q}$  on the tool (figure 5).

**Output.** The set of tool configurations  $(\mathbf{t}, \mathbf{R})$  for which  $\mathbf{p}$  and  $\mathbf{q}$  coincide, and which are collision-free in an environment  $R(S, \mathbf{p})$  of  $\mathbf{p}$  with an extension determined by  $R(T)$ . Further, a tolerance radius  $r(\mathbf{p}, \mathbf{t}, \mathbf{R})$  of approximation on  $S$  for the local accessible tool configurations  $(\mathbf{t}, \mathbf{R})$  is reported, within which the error between the surface and the tool is less than a given bound for every configuration is reported.

This version of the enumeration problem can be treated as follows.

If  $R(T)$  is completely on one side of the tangent planes at points of  $\mathbf{q} \in T$ , then the intersection tests can be restricted to those parts of  $R(S, \mathbf{p})$  which are on the tool access side of  $S$  of the tangent plane of  $S$  at  $\mathbf{p}$ .

**Case 1 (empty set):**

This happens for instance if  $S$  is convex in on  $R(S, \mathbf{p})$ . In this case, all candidate points are possible.

**Case 2 (nonempty set):**

*First approach (Point sampling):* If sample points on the workpiece are used for its representation, a brute-force point-in-tool test for sample tool locations may be sufficient. The tool may be preprocessed for efficient point-in-tool checking. The points to be tested are got by coordinate transforms.

*Second approach (Curvature analysis):* A necessary condition for local accessibility is that the normal curvatures of workpiece do not exceed the normal curvatures of tool. A test can be performed by comparison of the normal curvature plots of the tool and the surface for a given configuration. Precalculated hierarchical intervals of curvature plots of the tool might be used to speed up the search. For a spherical tool this test solves the problem.

*Third approach (Hierarchy of intervals):* The parameter domain of the tool is subdivided into a hierarchy of intervals. For every interval the swept space of the tool is precalculated. The swept spaces are tested for intersection with the workpiece starting with the coarsest one. The swept spaces with empty intersections are reported.

For a given tool contact point  $\mathbf{p} \in S^I$  and a given feasible tool configuration  $(\mathbf{t}, \mathbf{R})$ , the *error tolerance radius*  $r$  is defined by considering the region  $R$  of points on  $S^I$  which are closest to any of the points of  $R(T)$  with distance less than a given bound  $\varepsilon > 0$ . For points on  $S^I$  in a  $C^2$ -smooth region, this means that the distance to the produced surface in normal direction is less than  $\varepsilon$ . The smallest distance of a boundary point of the resulting region to  $\mathbf{p}$  is taken as  $r(\mathbf{p}, \mathbf{t}, \mathbf{R})$ .

## Discrete implementation

A difficulty with the local accessibility test on a mesh approximation of the workpiece surface is that the shape of the surface is not represented with the necessary precision. A representation of the surface is necessary which lets reconstruct the true shape with sufficient sampling accuracy at every vertex of the mesh. Examples of possibilities of achieving the requirement are

1. access to the *smooth CAD model*, e.g. represented by NURBS surfaces [Hos92],
2. a *mesh with an approximation routine*, which has been constructed using the approximation routine. The approximation routine fits a smooth surface into the mesh. The quality of the approximation is given by the error between the original surface and the fitted smooth surface.
3. a *mesh with a point density* which is significantly better than the tool size ("sampling theorem").

If the sampling is poor, we also may have a local reconstruction problem which might be ill-posed and has to be solved heuristically, for example by moving least squares fitting. This happens e.g. for low-resolution meshes.

The error tolerance radius  $r$  can be calculated heuristically as follows. A fan of curves (e.g. normal section curves) on the surface of  $T$  with the tool contact point  $\mathbf{q}$  as center is selected (figure 5). On each of those curves, the point  $\mathbf{s}$  closest to  $\mathbf{q}$  is determined whose distance from the workpiece surface  $S^I$  exceeds  $\varepsilon$ . Let  $\mathbf{r}$  be the point closest to  $\mathbf{s}$  and to the tool contact point  $\mathbf{p} \in S^I$ . Let  $r$  be the distance of  $\mathbf{r}$  from  $\mathbf{p}$  on  $S^I$ . The minimum  $r$  over all curves is taken as  $r(\mathbf{p}, \mathbf{t}, \mathbf{R})$ .

The curve point  $\mathbf{s}$  can be obtained by taking sampling points on the curve starting at  $\mathbf{q}$  and calculating the closest distance of every sampling point to  $S^I$ .

Possibly applying the concept of spatial coherence could be useful in order to speed-up the search. This means that searching at neighboring points is started with an initial configuration close to a solution at the predecessor.

An alternative approach could be using a modification of a milling simulation program. For example, the discrete displacement fields by Ayasse et al. [Ayasse02] may be modified for that purpose.

The local accessibility space may be represented by enumeration of the local accessibility status of a set of sample configurations. One possibility is to rasterize the parameter space of the configurations regularly, and store the tolerance radius in an array of dimension equal to the number of parameters. A 0-entry denotes the non-contact configurations. We call this representation a *discrete local accessibility map*.

## 4.2 Tool-Workpiece Collision Detection

The tool-workpiece collision detection has to be applied to the region of the workpiece surface which has not been subject of local accessibility analysis. Analogously to local accessibility, two versions exist: the existence problem and the freespace problem. The existence problem asks for the existence of a tool configuration with  $\mathbf{p}$  as a contact point in which the tool does not collide outside region  $R(S, \mathbf{p})$  defined in section 4.1 where it has been subject of the local accessibility investigations. It can be solved by terminating one of the collision detection procedures outlined below if a collision-free contact configuration is found.

### 4.2.1 The Freespace Problem

The freespace problem can be made precise as follows.

#### Freespace Problem

**Input.** A point  $\mathbf{p}$  on the workpiece, a region  $R(T)$  of contact candidate points  $\mathbf{q}$  on the tool, the region  $R(S, \mathbf{p})$  from the local accessibility problem.

**Output.** The set of tool configurations for which  $\mathbf{p}$  and  $\mathbf{q}$  coincide, and which are collision-free with respect to  $S^T - R(S, \mathbf{p})$ .

#### Discrete implementation

For solving the freespace problem, existent online collision detection algorithms can be used [Got96, Len99]. They usually preprocess the involved objects, here the workpiece and tool, into a data structure. Then the data structure is used for efficient intersection detection for arbitrary mutual locations of the

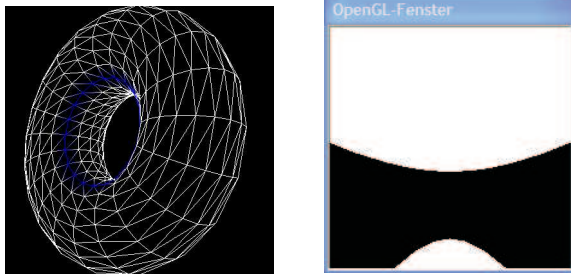


Figure 6: A visibility-based collision map (right) of a torus in the region indicated in blue color (left).

involved objects. We can proceed in this way due to the separation of local accessibility and collision. The reason is that, because of rounding errors, collision detection algorithms may report collision in the unavoidable contact case, too. However, if a collision detection algorithm reports the region of intersection in a sufficiently precise way, the contact case and the collision case including collisions in the region  $R(S, \mathbf{p})$  could be distinguished by analysis of its result.

As a pretest, visibility hulls may be used for rotationally modeled tools like ours. The investigated surface point serves as the viewpoint. A ray from the viewpoint defines the orientation of the tool axis by being in parallel to the ray. Sections of the visibility cone of a ray defined by back-and-front clipping planes are analyzed for being bounding boxes of sections of the tool. For tool sections where the visibility cone section is not a bounding box, a more detailed analysis becomes necessary.

The freespace can be stored in a *collision map* which has the same structure like the local accessibility map defined in section 4.1. Figure 6, left, shows a collision map represented as a raster image. The collision map is calculated for point located in the region indicated in blue on the surface of the torus in Figure 6, right. The collision test is simplified by testing rays of view through the points of a rasterized hemisphere located on the tangent plane on the exterior side of the torus. The black pixels represent colliding directions of the tool, that is, directions for which the ray intersects the surface, while the tool has free local access in the directions represented by the white pixels.

In general, the collision map can be calculated by applying a collision detection routine for every sampling configuration represented in the map. In the example of the visibility-based collision map collision detection reduces to the intersection of a ray with the surface.

A collision of a tool configuration, including local non-accessibility, may be represented by an entry 1 in the collision map, and feasible configurations by an entry 0. If the collision detection algorithm yields additional information about the safety of collision freeness, by e.g. reporting a closest distance between the tool and the workpiece outside the region  $R(S, \mathbf{p})$ , the single bit 0 in the collision map for non-colliding configurations may be replaced with a value between 0 and 1 expressing the safety of the configuration, a collision indicating by 1.

#### 4.2.2 Global accessibility

A contact point is called *globally accessible* if it can be reached from a basic configuration of rest of the tool and the machine along a tool movement path. Taking global accessibility into account is necessary since a configuration may be in contact at a workpiece point and collision free, but cannot be reached on a path.

The global accessibility problem can be defined as follows.

##### Global Accessibility Problem

**Input.** A point  $\mathbf{p}$  on the workpiece, a region  $R(T)$  of contact candidate points  $\mathbf{q}$  on the tool. the region  $R(S, \mathbf{p})$  from the local accessibility problem.

**Output.** The set of tool configurations for which  $\mathbf{p}$  and  $\mathbf{q}$  coincide, and which can be reached on a collision-free tool movement path from an initial configuration of rest of the tool and the machine.

##### Discrete solution

The task is checking every connected component of a collision map for whether one of its configuration  $c$  can be reached from the initial configuration of rest. It can be solved by a systematic discretized exploration based on simulation. For that purpose the configuration space may be rasterized e.g. in a regular way by

discretization of the domain of every control parameter. The exploration starts at configuration  $c$  and visits a sequence of neighboring discrete configurations. One possibility is a depth-first search guided by heuristic quality measure which forces the search in direction of the initial configuration of rest, for example according to the  $A^*$ -algorithm [Lat91]. Since just the existence of a path, but not its quality, is important, the quality measure has the main goal of restricting the search space. Multiple visits of configurations should be avoided by storing the already visited configurations, possibly in a compressing data structure like a hyper-octree.

A collision map, updated by removing those connected components which cannot be reached from the initial configuration of rest, is called a *feasible configuration map*. The configurations in its freespace components are called *feasible*.

Since the time and storage requirements may be considerable for connected components which cannot be accessed from the initial configuration of rest, those connected components for which these resources exceeds some time threshold without success could be removed, too, in practice, in order to achieve practical computing times.

#### **4.2.3 Collision-prevented accessibility constraint**

The collision-prevented accessibility constraint is given by the border between the set of workpiece points for which a collision-free contact between tool and workpiece exists, and the rest of the points.

By a proper definition of the region of interest, points which cannot be accessed because of collision should not occur. The collision-prevented accessibility constraint can be used to determine a region of interest.

#### **Discrete solution**

The existence of collision-free accessibility is determined for every vertex of the workpiece-representing mesh. The non-accessible vertices are removed from the mesh. The possibly resulting mesh boundary is the collision-prevented accessibility constraint.

In order to achieve a good approximation, the mesh might be refined in the environment of the border.

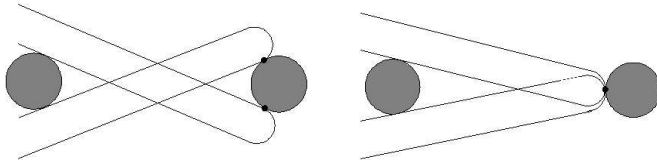


Figure 7: Example of a path on a torus with collision-critical points and corresponding tool configurations (left). The disks indicate a slice through the torus. On the right, a forbidden point on the path is shown.

### 4.3 Regularization

At a collision-critical point  $\mathbf{p}$  on a path, a collision may happen if the path contains a points  $\mathbf{q}$  in an environment of  $U(\mathbf{p})$  so that a freespace component  $C$  of  $U(\mathbf{p})$  exists which does not contain a feasible tool configuration of  $\mathbf{q}$ . If the tool is in  $C$  at  $\mathbf{p}$  it cannot reach  $\mathbf{q}$  without leaving  $C$  and thus being colliding.

However, a collision at a collision-critical point is not necessary. If the tool at  $\mathbf{p}$  is in a freespace component which  $\mathbf{p}$  shares with  $\mathbf{q}$  in an environment of  $\mathbf{p}$  on the path, the tool can move from  $\mathbf{p}$  to  $\mathbf{q}$  in this freespace component, and thus is collision-free.

Although every collision-critical point may be traversed locally without collision in a suitable configuration, it may happen that the whole path cannot be traversed in a collision-free way. An example are the circles induced by a cutting plane containing the rotational axis of a torus (figure 7). Such a circle contains two critical points which both can be locally traversed, but no collision-free path exists which has the closed circle as its translational curve.

A possibility of avoiding collisions is forbidding a number of points to be traversed by a path. Candidates for forbidden points are of course the critical points. However, forbidding all critical points is sometimes too much. In the torus example, for instance, it is sufficient to forbid just one of the critical points (figure 7). Even more, any other two-component point would be sufficient.

One approach to restricting the number of critical points is a method which we call regularization. *Regularization* means reducing the number of freespace components at every point of  $S^I$  using heuristic constraints, in the ideal case to just one (figure 8).

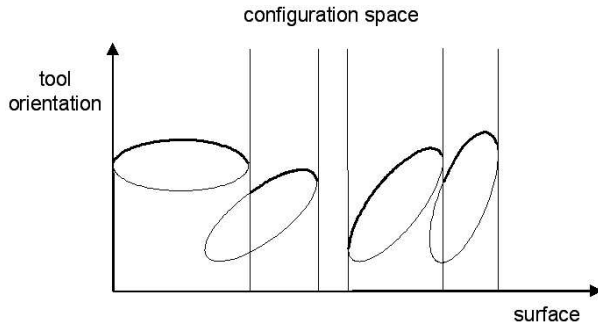


Figure 8: Configuration space restriction by regularization. The freespace components are indicated by the ellipse-shaped regions. The fat curves indicate the restriction achieved by regularization. The interval on the vertical axis over which no freespace region is located corresponds to singular points without any freespace configurations.

An example of regularization is regularization by quality functions. A quality function  $Q$  assigns a real number  $Q(\mathbf{q}, C)$  to every pair  $(\mathbf{q}, C)$ ,  $\mathbf{q} \in S^I$ ,  $C$  a freespace component at  $\mathbf{q}$ .  $Q$  has to be continuous at every point  $\mathbf{p} \in S^I$ , in the sense that an environment  $U(\mathbf{p})$  exists in which  $Q(\mathbf{q}, C)$  is continuous with respect to  $\mathbf{q}$  for all fixed freespace components  $C$  global in  $U(\mathbf{p})$ . Regularization at a point  $\mathbf{p}$  is performed by assigning those freespace components  $C$  at  $\mathbf{p}$  to  $\mathbf{p}$  for which  $Q(\mathbf{p}, C)$  has the minimum value among those global in some environment  $U(\mathbf{p})$ .

#### 4.3.1 Regular and Singular Points

In this way, two types of points emerge: the *regular* ones which have got a unique freespace component, and the *singular* ones which have still more than one freespace component, or none. No freespace component exists if none of the tool configurations at  $\mathbf{p}$  is an inner point of a freespace in the unrestricted tool configuration space (figure 8).

For a path using just configurations in the freespaces obtained by regularization, collision-freeness is given by definition at all regular points. Hence, if path

planning yields just paths without inner critical points, the resulting paths are safe.

An example of a quality function for regularization is the angle between the normal of  $S$  and the axis of the tool in a given configuration at a point  $\mathbf{p} \in S$ . This quality function prefers freespaces which allow tool configurations with minimal deviation from the normal which is reasonable for five-axis milling. In the torus example, the singular points lie on the inner equator of the torus. Hence we now have just one point instead of two before, which are candidates for collision. If path planning avoids traversal of the equator, the paths are safe.

A second example is the angle between the axis of the tool in its best access direction and the axis of the tool in a given configuration.

A third example of a quality function is the kinematic quality of the configurations of the production machine, or a combination of it with the access angle.

### 4.3.2 Regular and Singular Curves

Let  $\mathbf{k}$  be a curve connecting two points  $\mathbf{p}$  and  $\mathbf{q}$  on  $S^I$ . A freespace component  $C(\mathbf{p})$  of  $\mathbf{p}$  and a freespace component  $C(\mathbf{q})$  of  $\mathbf{q}$  are adjacent with respect to  $\mathbf{k}$  if the freespace of  $\mathbf{k}$  has a freespace component  $C(\mathbf{k})$  which contains a feasible configuration for every point on  $\mathbf{k}$  and whose restrictions on configurations feasible for  $\mathbf{p}$  and  $\mathbf{q}$ , respectively, are just  $C(\mathbf{p})$  and  $C(\mathbf{q})$ , respectively. This means that all configurations of  $C(\mathbf{q})$  can be reached from  $C(\mathbf{p})$  on a path whose workpiece contact curve is  $\mathbf{k}$ .

Under the assumption of a regularization of the freespaces, the following cases may occur:

1. *Both end vertices  $\mathbf{p}$  and  $\mathbf{q}$  are regular:* If the two unique freespace components  $C(\mathbf{p})$  and  $C(\mathbf{q})$  are adjacent, then  $\mathbf{k}$  is called *regular*. In this case a collision-free path exists between  $C(\mathbf{p})$  and  $C(\mathbf{q})$ .

If the two freespace components are not adjacent, then  $\mathbf{k}$  is called *singular*. In this case no collision-free path exists between the  $C(\mathbf{p})$  and  $C(\mathbf{q})$  whose workpiece contact curve is  $\mathbf{k}$ .

2. *At least one end vertex is singular:* In this case the curve is called *singular*, too. Although a collision-free path between feasible configurations at

$p$  and  $q$  in contact with  $k$  could exist, we do not try explicitly to take an advantage of it.

### **Discrete approximation**

For discrete solution, the surface curve is replaced with the straight line segment of an edge of the approximating workpiece mesh. An edge is called *regular* or *singular*, respectively, if conditions analogously to surface curves hold.

If the approximation by the straight line segment is within the accepted production tolerance, this approach is feasible. If the true surface is still available, a transfer of the path to this surface, and validation and correction by simulation, may be used for optimization. This holds in particular if the approximation by a straight line segment exceeds the production tolerance.

## **4.4 Extension of a Contact Curve to a Tool Movement Path**

By regularization, the search space for paths has been restricted. Under this restriction, both approaches to path planning yield a finite number of workpiece surface curves, together with a freespace component which contains at least one feasible configuration for every curve point. In a discretized representation, the workpiece surface curves are polygonal chains, with a freespace component at every vertex so that the freespace components of two adjacent vertices are adjacent.

In addition to the workpiece surface curves, movements of the tool not in contact with the workpiece may be necessary. The two motion planning approaches of this paper report pairs of vertices of a workpiece mesh between which a non-contact, or idle, movement is necessary as part of the overall tool movement part. In the next subsection we outline a possibility of constructing an environment curve in this case which is represented analogously to the contact curves. Subsection 4.4.2 describes a general approach to extracting a concrete tool movement path from those representations.

#### 4.4.1 Environment Curves

One approach to finding a favorable contact-free tool path is to find a polygonal chain  $\mathbf{q}_0, \dots, \mathbf{q}_n$ ,  $\mathbf{q}_0 := \mathbf{p}_i$  and  $\mathbf{q}_n := \mathbf{p}_{i+1}$  in the environment of the workpiece  $V$  and an assignment of a freespace component of the tool or machine configuration space to every vertex  $\mathbf{q}_j$  so that a collision-free transition is possible between two consecutive vertices of the chain. We call this an *environment curve*.

The vertices of the polygonal chain may be vertices of a spatial mesh in the environment of the workpiece mesh. The spatial mesh defines a volume fringe around the workpiece surface whose thickness has to guarantee the existence of a collision-free path so that a feasible tool path exists between any two points of the workpiece mesh whose tool center points are within the fringe. The contact between the workpiece surface and spatial mesh is established by defining the spatial mesh so that the workpiece mesh is a subset of the boundary of the spatial mesh.

To every vertex of the spatial mesh, a freespace component of the tool or machine configuration space is assigned. The freespace components of the vertices of the spatial mesh belonging to the workpiece mesh have the freespace components of the workpiece mesh. An edge of the graph is tagged as *feasible* if a collision-free transition is possible between the two end points on their connecting line segment. The subgraph induced by the tagged edges has to be connected.

One approach to constructing such a spatial mesh and graph is to take a 3d-grid graph starting near the surface of the workpiece, and connecting the vertices of the surface mesh to the grid vertices in their environment. To every vertex of the grid graph, the set of tool configurations is assigned having it as tool reference point. Starting at the surface mesh, the grid is processed in breadth-first strategy. To a not yet processed vertex, a connected component of the freespace is assigned to which a collision-free transition from the freespace component of an already processed neighboring vertex in the graph exists. All edges for which this holds are tagged. If more than one connected component is possible, one is chosen according to a heuristic regularization criterion. The breadth-first construction may stop when the subgraph induced by the tagged edges is connected.

It may be that this procedure fails. This could happen even if global accessibility

is satisfied for the connected components used on the surface mesh. Reasons could be missing transitions because of discretization and missing transitions because of regularization. An approach to remedy the first effect is finer discretization at critical locations. The second effect can possibly be overcome by assigning more than one connected component at the grid vertices. In this case a grid vertex is replaced with copies, one for every component, which are made incident by an edge to a neighboring vertex if a collision-free transition is possible between the corresponding components.

#### 4.4.2 Extraction of a Path

From the representation of contact and environment curves, one configuration has to be selected for every vertex of the polygonal chain, in order to get a concrete milling path. The configurations should be chosen so that an objective function  $G$  is minimized, which, for example, expresses fairness of tool and machine parts acceleration. In general, such an objective function has the form

$$G(c_0, \dots, c_n) = \sum_{i=0}^{n-k} H(c_i, \dots, c_{i+k})$$

where  $c_0, \dots, c_n$  are feasible configurations at the vertices  $\mathbf{p}_0, \dots, \mathbf{p}_n$  of the path. Each of the configurations  $c_j$  is an element of the freespace component assigned to  $\mathbf{p}_m$  of the discrete collision map of the vertex. In  $G$ , the quality of a path depends on the quality of all consecutive subpaths of length  $k + 1$ . A typical value is  $k = 2$  which is at least required if the variation of tool or machine acceleration is taken into account.

An approach to discrete path selection for this type of objective function is the algorithmic paradigm of dynamic programming. The calculation state at step  $t$  is expressed by tuples

$$(c_t, \dots, c_{t+k}, G_t(c_t, \dots, c_{t+k}))$$

with

$$G_t(c_0, \dots, c_k) = H(c_0, \dots, c_k)$$

for  $t = 0$  and with

$$G_{t+1}(c_{t+1}, \dots, c_{t+1+k}) = \min_{c_t} \{G_t(c_t, \dots, c_{t+k}) + H(c_{t+1}, \dots, c_{t+1+k})\}$$

for  $t > 0$ . By storing back-references to the arguments for which the minimum has been achieved, an optimal sequence of configurations can be reconstructed from the best among the solutions  $G_{n-k}(c_{n-k}, \dots, c_n)$ .

A further improvement might possibly be achieved if the candidate set is extended to further freespace components beyond the one resulting from regularization.

Since the tool movement path is calculated on a discrete approximation, it should be checked for collision freeness and quality by simulation. If troubles are recognized, a new trial of path planning could be started on a refined discrete approximation.

## 5 A Graph Path Covering Approach to Path Planning

The background of the graph path covering approach to path planning is the observation, that a disk of radius  $r$  moved on a finite set of paths and containing every vertex of a geometric planar graph with edge lengths less than  $r$  covers the region spanned by the graph without gaps. The disk stands for the tool, and the planar graph for the workpiece mesh. The best case would be just one path containing every vertex exactly once. A graph path of this property is called a *Hamiltonian path*, or a *Hamiltonian cycle* if its last vertex is connected with the first one by a graph edge. Graphs do not need to be Hamiltonian, i.e. having a Hamiltonian path. By augmenting a given graph by an edge between every pair of vertices not yet adjacent, a complete graph is obtained which always is Hamiltonian. A Hamiltonian path with a minimum number of edges not belonging to the original graph induces a minimum set of paths which cover every vertex in a disjoint way. In the following we extend this idea on the path planning problem on curved surfaces, controlling the feasibility and shape of the surface path by a objective function.

We start with an algorithmic framework for the graph path covering approach (section 5.1). The first step of the framework is finding an appropriate mesh which in particular must have edge lengths guaranteeing covering of the surface by the tool (section 5.2) within an error tolerance. The next issue addressed is the definition of the path objective function (section 5.3). Finally, algorithmic solutions to the calculation of Hamiltonian contact paths are presented (sec-

tion 5.4).

## 5.1 A Framework for Graph Path Covering Approaches

The graph path covering approach can be summarized as follows.

### Framework for graph path covering approaches to path planning

**Step 1 (Discrete covering)** Find a finite set of points on  $S^I$  for which a feasible tool configuration exists so that the remaining material lies within the given tolerance.

**Step 2 (Constraint setup)** Define constraints and objective functions related to constraints, required for feasible paths:

1. The path is crossing-free.
2. A path segment between two consecutive points is either completely on the surface and within an accessible region, or off the surface.
3. The path is collision-free.
4. The path is fair with respect to tool and machine acceleration.

**Step 3 (Contact path construction)**

**Step 4 (Freespace optimization and path extraction)** Optimize the assignment of freespace components and extract a concrete path of configurations from the freespace components.

In the following, the details are explained.

## 5.2 Error-adapted Meshing

Error-adapted meshing is the subject of step 1 of the algorithmic framework of section 5.1. Two types of errors have to be taken into account: the usual surface approximation error and the problem-specific production error.

For a surface-approximation-error-adapted triangulation, the distance between the mesh and the original surface has to be within a given tolerance, using a suitable distance function between surfaces.

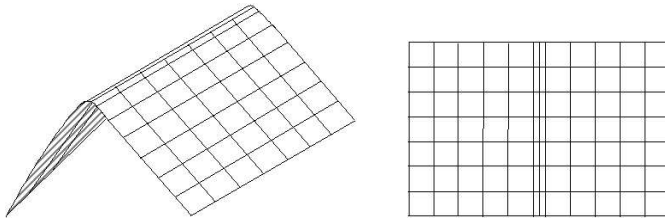


Figure 9: An ideal example of a production-error adapted mesh. The diagonals splitting the rectangles into triangles are omitted.

For production-error adapted triangulation, the sub-surface induced by the triangles incident to a vertex of the mesh has to be within the environment of the vertex on the surface defined by the error tolerance radius  $r$ . For simplification, we represent this condition by demanding that the lengths of the edges incident to the vertex are less than  $r$ . If the condition is satisfied, the covering property of the path induces that every point of the mesh, and, approximately, every point on the original surface, is in the  $r$ -neighborhood of the path.

Figure 9 shows an ideal example of a production-error adapted mesh. The diagonals splitting the rectangles into triangles are omitted. Figure 10 presents triangulations of a mesh which emphasize on optimal adaptivity of surface with a low number of triangles (left) and on "pretty" triangles with the goal of equal angles (right)[Had06]. The first alternative is evidently curvature-adapted and a good starting point for a production-error adapting refinement.

In our case, taking just the production error into account is sufficient since a good approximation of the mesh to the tool induces a good approximation to the workpiece surface.

## Implementation

An adaptive triangulation may be constructed for example in the following way, dependent on the input representation of the workpiece surface:

*for a smooth CAD model:* Application of an adaptive meshing routine which takes  $r$  as a bound on the edge lengths of the resulting mesh as stopping

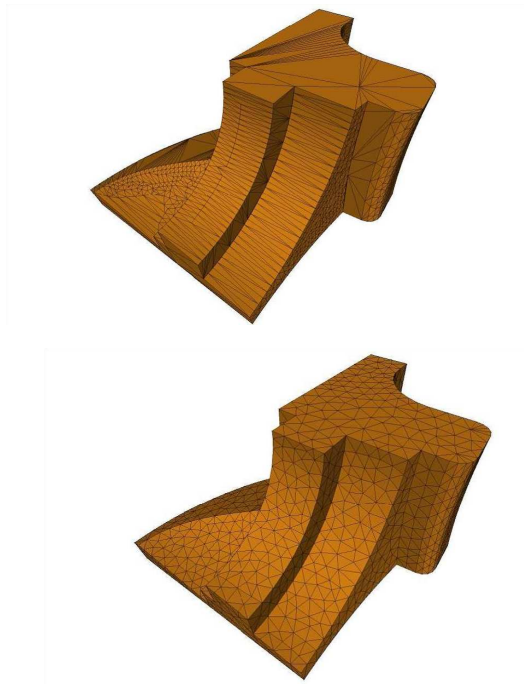


Figure 10: Triangulations of a mesh which emphasize on optimal adaptivity of surface with a low number of triangles (left) and on "pretty" triangles with the goal of equal angles (right). The first alternative is evidently curvature-adapted an a good starting point for a production-error adapting refinement.

criterion. An example is iterative subdivision of an initial coarse approximation like the approach by [Ruprecht et al.] with additional removal of "hanging" nodes ("T"-vertices).

*for a mesh with an approximation routine which has been constructed using the approximation routine:* This is similar to the preceding version, by using the given mesh as starting point for refinement.

*for a mesh with a point density which is significantly better than the tool size:* Application of an adaptive mesh reduction procedure, e.g. the one by

Kobbelt et al. [Kob00], with  $r$  as a bound on the edge lengths of the resulting mesh.

### 5.3 Path Objectives Function

The objective function for paths is required in step 2 of the algorithmic framework of section 5.1.

The path objective function controls the shape of a path. It is a weighted sum of sub-objective functions,

$$F(\mathbf{p}_1, \dots, \mathbf{p}_m) = w_a \cdot F_a(\mathbf{p}_1, \dots, \mathbf{p}_m) \quad (1)$$

$$+ w_n \cdot F_n(\mathbf{p}_1, \dots, \mathbf{p}_m) \quad (2)$$

$$+ w_t \cdot F_t(\mathbf{p}_1, \dots, \mathbf{p}_m) \quad (3)$$

$$+ w_l \cdot F_l(\mathbf{p}_1, \dots, \mathbf{p}_m) \quad (4)$$

where

$F_a$  evaluates the derivation of the curvature,

$F_n$  evaluates the normal curvature,

$F_t$  evaluates the tool orientation and collision,

$F_l$  evaluates the non-contact tool movements, and

$w_a, w_n, w_t, w_l$  non-negative weights summing up to 1.

#### 5.3.1 Derivation of the curvature

If an arc-length parametrization is chosen, as we do, the acceleration and curvature coincide. In the discrete case of a polygonal chain  $(\mathbf{p}_0, \dots, \mathbf{p}_m)$  as a curve, the following holds (figure 11):

**speed:** The vector between two points, divided by its length:

$$\mathbf{v}_i := \mathbf{v}(\mathbf{p}_i) := \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}$$

$$v_i := \|\mathbf{v}_i\| = 1.$$

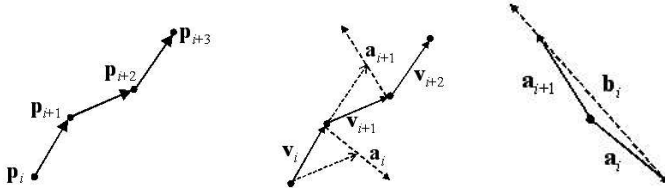


Figure 11: Illustration of the discrete approximation of speed, acceleration, and derivative of acceleration.

**acceleration:** The difference vector of the vectors of speed, divided by length:

$$\mathbf{a}_i := \mathbf{a}(\mathbf{p}_i) := \frac{\mathbf{v}_{i+1} - \mathbf{v}_i}{\|(\mathbf{p}_{i+1} - \mathbf{p}_i)\|}$$

$$a_i := \|\mathbf{a}_i\|.$$

**derivation of acceleration:** The difference vector of acceleration vectors divided by length:

$$\mathbf{b}_i := \mathbf{b}(\mathbf{p}_i) := \frac{\mathbf{a}_{i+1} - \mathbf{a}_i}{d(\mathbf{p}_{i+1}, \mathbf{p}_i)}$$

$$b_i := \|\mathbf{b}_i\|.$$

The derivation of the acceleration satisfies  $0 \leq b_i \leq 2$ .

The immediate application of the definitions does not yield reasonable results because of the discretization effect of the triangulation. We reduce the difficulties by curve smoothing, and have chosen the polygon reduction algorithm by Imai and Iri [Ima86] for this purpose (figure 12). In addition to the points  $\mathbf{p}_i, \dots, \mathbf{p}_{i+3}$  we consider additional  $s \geq 0$  points before and after  $\mathbf{p}_i$  and  $\mathbf{p}_{i+3}$ , respectively. According to the Imai/Iri algorithm, a graph  $G = (V, E)$  is constructed. The vertex set  $V$  is defined by the considered points,  $V := \{\mathbf{p}_{i-s}, \dots, \mathbf{p}_{i+3+s}\}$ . The edge set consists of all pairs  $(\mathbf{p}_{i_1}, \mathbf{p}_{i_2})$  of vertices in  $V$  so that a geodesic shortest path on the mesh surface between  $\mathbf{p}_{i_1}$  and  $\mathbf{p}_{i_2}$  runs within the triangles incident to the polyline defined by the vertices of  $V$ . A geodesic path on the mesh is a shortest path if the mesh is considered as surface, that is, it may traverse the interior of the triangular facets, too. The incident triangles define a tolerance region whose extension depends on the

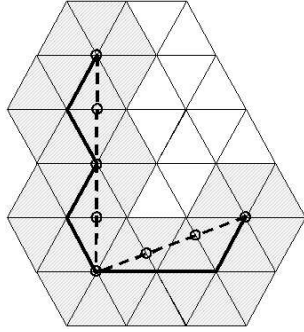


Figure 12: Projection of vertices of the original graph path onto the polygonal path resulting by curve smoothing.

sampling density of the mesh. According to the Imai/Iri algorithm, a shortest path between  $\mathbf{p}_{i-s}$  and  $\mathbf{p}_{i+3+s}$  yields the desired smooth approximation. It is a polyline whose vertices are induced by the traversed mesh vertices and edges.

An assignment between the original path and its smooth approximation is calculated by assigning a vertex  $\mathbf{p}_{i_j}$  of the original path to vertex  $\tilde{\mathbf{p}}_{i_j}$  of the new path which is closest along a mesh edge (figure 12). The values  $\mathbf{v}_i$ ,  $\mathbf{a}_i$ , and  $\mathbf{b}_i$  are calculated on the path segment between  $\tilde{\mathbf{p}}_i$  and  $\tilde{\mathbf{p}}_{i+2}$ . If it contains additional vertices besides the assigned ones, the values are evaluated on every subpath-segment of four consecutive vertices and their maximum is taken.

In [Sto05], a slightly different measure has been defined. It considers the angles between  $e_i$  and  $e_{i+1}$ , between  $e_{i+1}$  and  $\bar{e}_{i+2}$ , between  $e_i$  and  $\bar{e}_{i+2}$ , and between  $\bar{e}_{i+2}$  and  $e_{i+2}$ .  $\bar{e}_{i+2}$  denotes the orthogonal projection of the line segment  $e_{i+2}$  onto the plane spanned by  $e_i$  and  $e_{i+1}$ . From each of those angles, a scalar value between 0 and 2 is derived, and the resulting values are summed up yielding a quality value  $\tilde{b}_i$ . A possible advantage of  $\tilde{b}_i$  could be that the components of  $\bar{e}_{i+2}$  on the plane of  $e_i$  and  $e_{i+1}$  and the one perpendicular to it, respectively, are separated. They could be taken into account with different weights in the sum of  $\tilde{b}$  in order to possibly have more influence on the desired behavior of the curve. However, this additional flexibility has not been used yet.

The sub-objective function representing the derivation of the curvature is defined as

$$F_a(\mathbf{p}_1, \dots, \mathbf{p}_m) := \sum_{i=0}^{m-3} b_i \cdot \|\mathbf{p}_{i+1} - \mathbf{p}_i\|.$$

The factor  $\|\mathbf{p}_{i+1} - \mathbf{p}_i\|$  weights the contribution of an edge relative to the path length.

### 5.3.2 Normal curvature

A path following a line of minimal or maximal curvature on a surface has no curvature component in a direction of the tangent plane, that is, if the geodesic curvature is 0. While the effect of normal curvature is not avoidable because it depends on the degree of non-flatness of a surface, the geodesic curvature can be controlled. As already mentioned earlier, the normal curvature can also be used as a criterion on the orientation of a path concerning the production error. These observations show that it is useful to have possibility of normal curvature control in the objective function of a path.

For this purpose, a quality value  $c_{\min,i}$  (and analogously  $c_{\max,i}$ ) is defined for every edge  $e_i$ . It uses a vector  $\mathbf{k}_{\min,i}$  which gives the directions of minimal curvature, respectively, of the workpiece surface at point  $\mathbf{p}_i$ . Let  $\alpha_{\min,i}$  be the smaller of the two angles between the straight line in direction  $\mathbf{k}_{\min,i}$  through  $\mathbf{p}_i$  and the line induced by  $e_i$ . Then

$$c_{\min,i} := \min\{\alpha_i, \alpha_{i+1}\}.$$

The reason for taking the minimum and not the sum or the maximum of  $\alpha_i$  and  $\alpha_{i+1}$  is encouraging the path finding algorithm to approach the preferred direction of motion in vertices in which a larger deviation between the current and desired direction is given.

The sub-objective function representing the normal curvature criterion is defined as

$$F_n(\mathbf{p}_1, \dots, \mathbf{p}_m) := \sum_{i=0}^{m-1} c_{\min,i} \|\mathbf{p}_{i+1} - \mathbf{p}_i\|.$$

### 5.3.3 Tool orientation and collisions

Examples for ratings of the quality of the tool orientations are

1. the angle between the normal of  $S$  and the axis of the tool in a given configuration at a point  $\mathbf{p} \in S$ ,
2. the angle between the axis of the tool in its best access direction and the axis of the tool in a given configuration,
3. the kinematic quality of the configurations of the production machine, or a combination of it with the access angle.

Let  $\tau_i$  be the value of one of those measures at a point  $\mathbf{p}_i$  for some tool configuration. For an edge  $e_i$  between two points  $\mathbf{p}_i$  and  $\mathbf{p}_{i+1}$  with corresponding tool configurations, a quality value for the tool orientation behavior is

$$o_i := (\tau_i + \tau_{i+1} + |\tau_i - \tau_{i+1}|).$$

The first two terms of the first factor represent the quality of the configuration at the two end points, while the third term takes into account their variation.

Since the approach of path covering uses surface paths and not tool motion paths in the configuration space, we have to eliminate specific tool configurations in the quality measure. We achieve this goal by regularization (section 4.3), that is, we choose a configuration at the vertices  $\mathbf{p}_i$  and  $\mathbf{p}_{i+1}$ , respectively, which optimizes  $\tau_i$  and  $\tau_{i+1}$  under the constraint of collision-freeness. This approach is possible if both vertices and the edge itself are regular.

In the case of singularities (cf. section 4.3.2), the quality value should be high in order to give the edge low priority during path search. If for at least one of the endpoints no feasible tool configuration exists, then the edge will be considered in the part  $F_l$  of the objective function (section 5.3.4), and gets a value of  $o_i = 0$  in  $F_t$ . Otherwise a configuration of best quality is chosen in the freespace components selected by regularization at  $\mathbf{p}_i$  and  $\mathbf{p}_{i+1}$ , respectively. Let  $\tau_{i,i+1}$  be the angle between the axes of the two configurations. Then we set

$$o_i := (\tau_i + \tau_{i+1} + |\tau_i - \tau_{i+1}|)\exp(\tau_{i,i+1}).$$

Big values of  $\tau_{i,i+1}$  can for example be observed if the tool has to access from different sides relative to an obstacle, as shown in figure 7. By the additional

exponential factor the chance increases that singular edges will not be in the resulting path. However, this is not guaranteed so that a step of verification of collision freeness of the resulting path has to be performed during path construction.

In [Has04], a quality measure based on a more detailed parametrization has been proposed, but the effect of both measures should be similar.

The sub-objective function representing tool motion and collision is defined as

$$F_t(\mathbf{p}_1, \dots, \mathbf{p}_m) := \sum_{i=0}^{m-1} o_i \cdot \|\mathbf{p}_{i+1} - \mathbf{p}_i\|.$$

### 5.3.4 Non-contact Tool Movements

It may happen that a tool path on which the tool is always in contact with the workpiece surface does not exist, or is less suited than a path on which the tool is lifted off at some points and set down at a next one not in its neighborhood. The path segment in-between those two points is a *non-contact tool movement*. The non-contact path segments could be of different complexity with respect to execution time and efforts by the production machine. The complexity can be taken into account by the component  $F_l$  of the objective function.

Mesh edges for which both endpoints have a feasible tool configuration, are not taken into account in  $F_l$ . They are represented by a value  $l_i = 0$ .

For non-mesh edges  $\mathbf{p}_i$  and  $\mathbf{p}_{i+1}$ , a weight  $l_i$  is chosen which expresses the complexity of the possible non-contact paths between the two points. The weight may be chosen by a crude estimation of the complexity, for example the Euclidean distance of  $\mathbf{p}_i$  and  $\mathbf{p}_{i+1}$  in the space, or the geodetic distance of  $\mathbf{p}_i$  and  $\mathbf{p}_{i+1}$  on the mesh.

A more precise, but rather time consuming approach could be the calculation of a favorable contact-free tool movement path between  $\mathbf{p}_i$  and  $\mathbf{p}_{i+1}$  according to section 4.4.1, and deriving a weight  $l_i$  from it. The weight could be its length in the tool or machine configuration space according to some suitable metrics.

The sub-objective function controlling contact-free tool motion is

$$F_l(\mathbf{p}_1, \dots, \mathbf{p}_m) := \sum_{i=0}^{m-1} l_i \cdot \|\mathbf{p}_{i+1} - \mathbf{p}_i\|.$$

## 5.4 Calculation of Hamiltonian Contact Paths

The calculation of Hamiltonian contact paths is performed in step 3 of the algorithmic framework of section 5.1.

The quality values of the preceding section have the property that they are local in the sense that they are not dependent on the behavior of a whole path. The values  $c_{\min,i}$ , and  $o_i$  depend only on data available at the end points of the edge and properties of the edge, but not on the preceding and subsequent edges on the path. Hence the values, multiplied by  $\|\mathbf{p}_{i+1} - \mathbf{p}_i\|$ , can be assigned to every edge independent from a concrete path. The evaluation of the objective function of a path consists in adding the values assigned to the edges of the path.

Analogously, the value  $b_i$  depends on a path of  $4 + 2s$  consecutive edges, the rest of the possibly longer path is not relevant. This property allows to assign a value  $b_i \cdot \|\mathbf{p}_{i+1} - \mathbf{p}_i\|$  to every subpath of the mesh defined by  $4 + 2s$  vertices (section 5.3.1).

The goal is finding a Hamiltonian path so that the objective function is minimized. However, the existence of a Hamiltonian path on the mesh is not guaranteed. For that reason the mesh graph is extended to a complete graph by inserting edges between all pairs of vertices not connected by an edge mesh. The weights of the new edges are high in comparison to the mesh edges in order to prevent the algorithm to make too much use of them.

With respect to the edge weights  $c_{\min,i}$  and  $o_i$ , the task is to find a traveling salesman roundtrip. A traveling salesman roundtrip is a Hamiltonian path for which the sum of the weights of the edges is minimum. The traveling salesman problem is a very well investigated problem [Law85, Jue97]. Although it is NP-hard [Gar79], powerful algorithms exist for its exact solution on non-trivial graph sizes as well as heuristics yielding good approximations to the optimum. In [Has04], the Lin-Kerningham-heuristics [Lin73] with improvements by Helsgaun [Hel00, Hel02] have been applied.

When taking  $b_i$  into account, the problem to be solved is not the conventional traveling salesman problem. In particular it seems that the successful heuristics by Lin-Kerningham and Helsgaun cannot be transferred immediately to this modified problem type. Arkin et al. [Ark01] have treated this modified problem in the quite special version of covering paths on regular grid graphs in the plane with the goal of minimizing U-turns. Their approach cannot be applied in or more general setting.

Another aspect is that the requirements of  $b_i$  are not conform with those of the other two objectives which leads to the issue of a multicriterial optimization problem with criteria which cannot be simultaneously optimized [Ste86]. One way of approaching multicriterial optimization problems is to calculate so-called Pareto sets. A Pareto set is a set of solutions which do not dominate each other. A solution dominates a second one if it is not worse with respect to all involved criteria.

#### 5.4.1 Calculation by Evolutionary Algorithms

Given this situation we propose to apply evolutionary-type algorithms of computational intelligence to our Hamiltonian path problem.

Their basic idea of evolutionary computing is considering elements of the solution space of an optimization problem as individuals [Mic92, Bae00, Zit03]. A population of individuals is maintained which is modified by operations like mutation, recombination, and selection. The goal is finding a population which contains an optimum solution. The individuals are evaluated by a fitness function which is the basis of selection. Several implementations of this general idea exist, like e.g. genetic algorithms and evolutionary strategies.

Evolutionary algorithms have also been applied to the classical traveling salesman problem [Gre85, Gol85].

#### Representation of individuals

The individuals are the roundtrips. They are represented by their sequence of vertices in the order of the visit. The data structure is an integer array of size equal to the number of vertices. The  $i$ -th array element refers to the  $i$ -th vertex of the roundtrip. Furthermore, the consecutive vertices are arranged in a doubly linked list.

#### Operations of mutation

Mutation can be performed by the following two operators:

***k*-opt operator:** The  $k$ -opt operator [Bae00],  $k \geq 2$ , chooses  $k$  vertices  $\mathbf{v}_{i_1}, \dots, \mathbf{v}_{i_k}$  on the roundtrip (figure 14). This divides the roundtrip

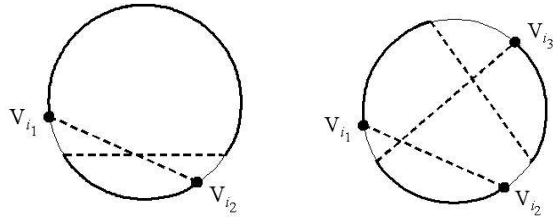


Figure 13: Illustration of the 2-opt and the 3-opt operator. The circle indicates a given roundtrip, the chords the newly inserted edges.

into subpaths  $\mathbf{v}_{i_j}, \dots, \mathbf{v}_{i_{j+1}}$  between two consecutive chosen vertices. A new individual is obtained by selecting any number of subpaths, reverse them, and connect them in the old order. Reversing the subpath  $\mathbf{v}_{i_j}, \dots, \mathbf{v}_{i_{j+1}}$  means replacing the old roundtrip edge  $(\mathbf{v}_{i_j}, \mathbf{v}_{i_{j+1}})$  with  $(\mathbf{v}_{i_{j+1}}, \mathbf{v}_{i_j})$ . Furthermore, the edge  $(\mathbf{v}_{i_{j+1}}, \mathbf{v}_{i_{j+1}+1})$  is replaced with  $(\mathbf{v}_{i_j+1}, \mathbf{v}_{i_{j+1}+1})$ . If the subsequent subpath has to be inverted, this new edge  $(\mathbf{v}_{i_j+1}, \mathbf{v}_{i_{j+1}+1})$  is used instead of  $(\mathbf{v}_{i_{j+1}}, \mathbf{v}_{i_{j+1}+1})$ . For a closed roundtrip, this procedure is applied cyclically.

**Lin-Kernighan operator:** A difficulty with the  $k$ -opt operator is the constant value of  $k$ . The Lin-Kernighan operator citeLin73 makes  $k$  variable. It constructs two edge sets  $X$  and  $Y$  of equal size, starting with the empty set and iteratively adding an edge  $x_j$  and  $y_j$ , respectively, in every step  $j = 1, \dots, k$ . The edges of  $X$  belong to the current roundtrip, while those of  $Y$  do not. After finishing  $X$  and  $Y$ , the edges of  $X$  are replaced in the tour with the edges of  $Y$ . The edges have the form  $x_j = (\mathbf{v}_{i_{2j-1}}, \mathbf{v}_{i_{2j}})$  and  $y_j = (\mathbf{v}_{i_{2j-1}}, \mathbf{v}_{i_{2j+1}})$ , where  $\mathbf{v}_{i_l}$  are vertices of the given path fulfilling the following constraints.

**sequential exchange:** The edges  $x_j$  and  $y_j$  have the same start vertex, and the edges  $y_j$  and  $x_{j+1}$  have the same end vertex. Further, the end vertex of the last edge  $y_k$  is the end vertex of the first edge  $x_1$ .

**feasible solution:**  $x_j = (\mathbf{v}_{i_{2j-1}}, \mathbf{v}_{i_{2j}})$  must be chosen so that its replacement with  $y_j = (\mathbf{v}_{i_{2j-1}}, \mathbf{v}_{i_{2j+1}})$  yields a roundtrip. An exception is the second iteration  $j = 2$  where the roundtrip is divided into two

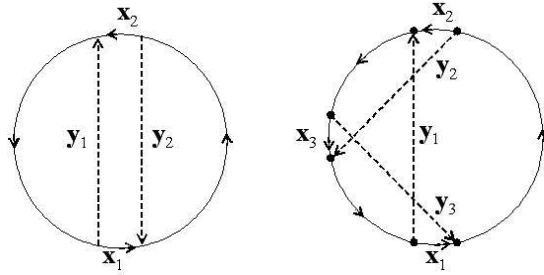


Figure 14: Illustration of the Lin-Kernighan operator for two edges and three edges.

smaller cycles. By fulfilling the condition in the next iteration the roundtrip becomes again feasible.

**positive gain:**  $y_j$  has to be chosen so that the gain of fitness achieved by replacing  $x_j$  by  $y_j$  should be positive.

**restriction of mutations:** The end vertex of a new edge  $y_j$  is among the  $k$  weight-nearest neighbors of its start vertex. In the original Lin-Kernighan algorithm further candidate sets based on the minimum spanning tree are used which speed up calculation significantly. However, the heuristics used are not favorable for optimizing  $b_i$  so that they are not applied here.

### Operations of recombination

Let  $I_1$  and  $I_2$  be two individuals. Their successor  $I$  obtained by recombination is constructed as follows:

1. Choose a subpath  $v_j, \dots, v_{j+k \bmod m}$  of  $I_1$  and initialize  $I$  with the subpath.
2. Augment  $I$  successively by vertices and edges, where  $v$  is the currently last vertex of  $I$ :
  - (a) If an edge of  $I_2$  with start vertex  $v$  and an end vertex not yet in  $I$  exists, then append the edge to  $I$  else

- (b) if the previous property holds for an edge of  $I_1$  then append the edge to  $I$  else
- (c) append an edge  $(\mathbf{v}, \mathbf{w})$  where  $\mathbf{w}$  is a vertex of  $I_2$  which has the shortest weight distance to  $\mathbf{v}$ .

This algorithm combines a complete subpath of  $I_1$  with subpaths of  $I_2$  which are as connected as possible.

### Selection by the plus-strategy

The plus-strategy generates a sequence  $\Phi_j, j = 1, \dots$ , of populations, each of a given size  $p$ , starting with an initial population  $\Phi_0$  of size  $p$ . It is parametrized, besides  $p$ , by two values,  $\mu \geq 0$  and  $\lambda \geq 0$ , with  $\lambda \leq \mu$  and  $\mu + \lambda \geq p$ .

The first step is selecting a set  $\Psi$  of  $\mu \leq p$  individuals from  $\Phi_j$ .  $\Psi$  is the set of input for the next step, recombination. Selection is performed by the so-called fitness-proportional strategy which will be explained later.

Recombination randomly selects two individuals  $I_1$  and  $I_2$  from  $\Psi$  and applies a recombination operation to  $I_1, I_2$  and  $I_2, I_1$ , respectively, assuming that the operation is not symmetric. The two new individuals are inserted in a set  $\Omega$ , and  $I_1$  and  $I_2$  are removed from  $\Psi$ . Recombination is repeated until  $\Omega$  contains  $\lambda$  individuals.

$\Omega$  is the input of the step of mutation. The individuals of  $\Omega$  are mutated by applying a mutation operator to every element of  $\Omega$ .

The last step of an iteration is taking the union of  $\Phi_j$  and  $\Omega$ , and selecting the  $p$  best individuals from it which together define the population  $\Phi_{j+1}$  as input for the next iteration.

The iteration stops after a time limit or if no further significant change of the fitness of the best individual in  $\Phi_j$  can be noticed. The best individual of the final population is reported as result.

The fitness-proportional strategy of selection used in the first step of the algorithm prefers elements of good fitness but every individual has a chance of survival. A probability of selection,

$$p(I) := \frac{\sum_{J \in \Phi} f(J) - f(I)}{(|\Phi| - 1) \cdot \sum_{J \in \Phi} f(J)},$$

is assigned to every element  $I$  of a population  $\Phi$ . Selection of an individual with this probability is implemented using the roulette wheel algorithm [Bae00]. The roulette wheel algorithm generates a uniformly distributed random number  $u \in [0, 1]$ , sums up the  $p(I)$  as long as the sum is less than  $u$ , and reports the last  $I$  of the resulting sum.

### **Selection by simulated annealing**

Simulated annealing applies a sequence of mutations, starting with an initial individual. If the fitness of the next individual  $I^+$  is improved, then the old individual  $I$  is replaced with the new one. If the fitness has become worse, then the new individual is accepted with probability  $p(I, I^+, T) := \exp(-|f(I^+) - f(I)|/T)$  where  $f$  is the fitness function.  $T > 0$  is a temperature which, starting with an initial value  $T_0$ , is successively reduced during calculation. In this way, the probability that a worse individual is accepted decreases, so that the algorithm finally converges.

Reduction of temperature is controlled by two parameters, the reduction rate and the reduction factor. The *reduction rates* determines the number of iterations after which the temperature is reduced next time. When a temperature-reducing iteration is achieved, the temperature is reduced by multiplication with the *reduction factor*.

### **Initial population**

The individuals of the initial population can be either generated randomly or by previous optimization. If the initial population has to have more than one element, a possibility is using the required number of copies of one individual.

### **5.4.2 Experiences**

The algorithms have been implemented based on quality measures similar to those presented in the preceding sections. One implementation did not take into account the fairness of curvature, in this way yielding a traveling salesman problem which is approximately solved with the Lin-Kernighan-Helsgaun approach. The other two implementations have used a criterion which aims on the

reduction of U-turns involving three consecutive edges, or four consecutive vertices, respectively. Since this version is not of the traveling-salesman-type, the evolutionary plus-algorithm and simulated annealing has been applied to this case.

### **Evolutionary plus-approach**

For the implementation of the described evolutionary plus-algorithm described in [Sto05], we have made the following observations.

1. With Lin-Kerninghan operator less iterations are required than with the  $k$ -opt operator in order to achieve similar results. Further, the computing time of the  $k$ -opt operator was higher.
2. A parameter of the Lin-Kerninghan operator is the number of iterations, i.e. the number of edges that should be maximally replaced. It has turned out that 10 is a good choice with respect to the quality of the results and the required computing time.
3. Recombination does not seem to have big influence on the quality of the results since without recombination very good results could be achieved, too. The fraction of fitness-improving recombinations among all trials has been between 5% and 10%, while 10% to 30% could be observed for mutations. A parameter controlling the probability of recombination has been introduced. If recombination does not take place, then the two individuals selected for recombination are taken over into the result without modification. We have chosen 0.2 for the recombination probability.
4. A population size of 50 has required about 30 minutes calculation time on a mesh of 256 vertices in order to get a favorable result. For population size 10, about 1 minute is required. A PC with a 0.8 GHz Pentium 3 processor and 96 MB RAM has been used.
5. Fitness-proportional selection has turn out to be quite time consuming. Explicit selection could be avoided by setting the parameters in a suitable way.
6. For the initialization of a population, two alternative approaches have been used. The first one, *random candidate*, starts with a random vertex

and successively adds vertices adjacent on the mesh in a random way, as long as possible. If not possible, any randomly chosen vertex not yet used is appended along a non-mesh edge.

The second approach, *random nearest candidate*, prefers vertices among the  $k$ -nearest to the currently last vertex.

7. For restriction of mutation,  $k = 10$  has turned out to be a useful value.

### **Simulated annealing**

For mutation the same as for the evolutionary plus-algorithm holds. Besides using a population of size 1, populations of bigger size, e.g. 10, have been used, too. It has turned out that the parameters of initial temperature, reduction rate, and reduction factor have to be chosen individually dependent on the input instance. A general setting seems not possible.

## **6 An Iso-potential Lines Approach to Path Planning**

The idea of the iso-potential lines approach to path planning can be explained as follows. Let us consider a simple polygonal region in the plane. To every point in the interior we assign the minimum distance of the point to any point on the boundary of the region. In this way we get a real function  $d$  over the polygonal region. This function is called distance function [Fri00, Per01]. Let  $c > 0$  be a real number less than the maximum  $d_{\max}$  of  $d$ . The points  $\mathbf{p}$  in the region with  $d(\mathbf{p}) = c$  define an implicitly represented curve which usually consists of one or more closed components. A curve of this type is called isoline or isocurve. The curves obtained for different values of  $c$  are nested. If  $d$  is considered as a mountain, the isolines represent the height lines indicating the height in a map.  $c$ -values on an equi-distant subdivision of the interval  $[0, d_{\max}]$  yield a family of isolines of visually uniform distance. If the interval size is less than the radius  $r$  of a disk, the disk movement along the isolines and the boundary of the polygon regions covers every point of the polygonal region. By connecting the stacks of nested isolines, a set of spiral-shaped curves is obtained which may serve a surface contact curves of a tool path.

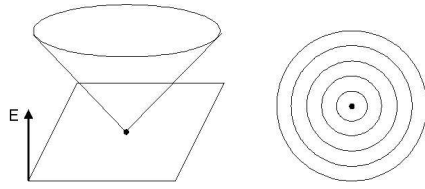


Figure 15: The distance function of a single point (left) and a corresponding set of isolines (right).

The chapter starts with an introduction of the concepts of the approach, like potential functions for which the distance function is a canonical example, isolines, control sets, and potential scaling (section 6.1). Then the algorithmic framework of the iso-potential lines approach is presented (section 6.2). In the following sections, several steps of the framework are described in detail: choice of a covering-generating control graph (section 6.3), adaptation of isolines to the accessibility error (section 6.4), potential and isoline smoothing (section 6.6), and concatenation of isocurves and arrangement of curve segments (section 6.7).

## 6.1 Potential Functions, Isolines, Control Sets, and Potential Scaling

The potential  $E$  is a nonnegative continuous scalar function on a region of interest  $S^I$  of the workpiece surface. The set of all points  $\mathbf{p} \in S^I$  with  $E(\mathbf{p}) = c$  for a constant number  $c > 0$  is called an isoline. If  $E$  is never constant on a full-dimensional subset of  $S^I$ ,  $E$  consists of curve segments including possibly isolated points.

We use potential functions emerging from distances defined on the surface  $S^I$ . An example is the geodesic distance which is available if  $S^I$  is sufficiently smooth. Intuitively, the geodesic distance between two points  $\mathbf{p}, \mathbf{q} \in S^I$  is the length of the shortest surface curve between  $\mathbf{p}$  and  $\mathbf{q}$  where the length of a curve is defined in the usual differential-geometric way. The distance function of a set  $M \subseteq S^I$  is a nonnegative scalar function  $D[M]$  on  $S^I$  which assigns to every point  $\mathbf{p} \in S^I$  the minimum distance  $D[M](\mathbf{p})$  to any point of  $M$ . Figure 15

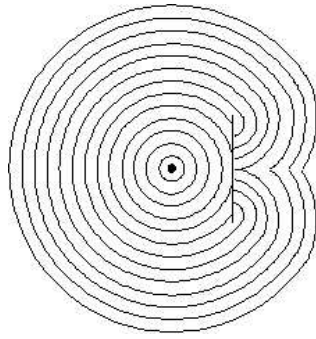


Figure 16: The isolines induced by a point as potential source and a line segment as obstacle.

shows the distance function and corresponding isolines of a single point in the plane, while figure 16 displays isolines of the distance function of a single point and a line segment as obstacle.

The distance functions  $D$  are potential functions. We call  $M$  the source of the potential  $D$ . The shape of  $D$  is controlled by  $M$ .

Let  $N \subseteq S^I$  be a further set. The distance function  $D[M, N]$  on  $S^I - N$  is called a distance function with obstacles where  $N$  defines the obstacles.  $D[M, N]$  is a potential function on  $S^I - N$ .  $N$  is another possibility of controlling the shape of  $D$ .

Let  $s$  be a positive real continuous function on  $S^I - N$ . The function  $D[M, N, s]$  which assigns to a point  $\mathbf{p} \in S^I - N$  the smallest value a curve integral of  $s$  along a curve from a point of  $M$  to  $\mathbf{p}$  may have, is called *scaled distance*. If  $s$  is equal to 1, then  $D$  remains unmodified. In a region where  $s > 1$ , two neighboring isolines become closer to each other, if  $s < 1$  their distance grows.  $s$  is another possibility to control the shape of a potential function.

The iso-potential lines approach to path planning uses a control graph  $G$  of curves on  $S^I$  as potential sources and obstacles. The *control graph* consists of finitely many vertices which are points on  $S^I$ , and a finite number of nonintersecting curve segments called control lines and either located between pairs of vertices, or being single closed loops or single curve segments. That is, the

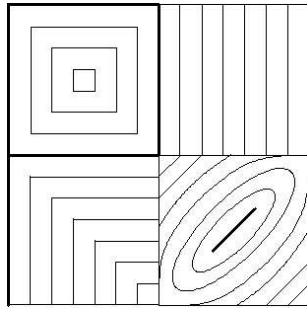


Figure 17: Example of a covering generating graph. Its edges are the edges of the squares and the fat line segment in the square on the bottom right.

graph is planar on  $S^I$ . The graph needs not to be connected and may have cycles. The connected components of  $S^I - G$  are called *surface segments*. The surface segments can be processed independently from each other.

Every curve segment of  $G$  is either a potential source or an obstacle. In order to have a potential value available at every point of  $S^I$  except on obstacles, every surface segment has to be adjacent to at least one potential source segment. In this case,  $G$  is called a *covering-generating control graph*.

### Discrete implementation

The control lines are defined by a subgraph of the workpiece mesh, preferably by a path.

The potential function is represented by a potential value at every vertex of the mesh. Additionally, the closest potential source vertices are stored at every vertex.

The scaling function is given by a scaling value assigned to every edge. The distance value of a vertex is the length of a shortest path (taking into account the edge weights) on the mesh to all potential source vertices which does not traverse any control vertex. One possibility to implement this is virtually cutting the mesh along the edges of the control graph.

The distance values can be calculated in a breadth-first way according to the

shortest path algorithm by Dijkstra. The closest potential source vertices can be found by storing back links to the vertices from which the optimal result at every vertex has been found.

The isopotential lines are constructed with a marching triangle algorithm, analogously to the marching squares algorithm [Cli88].

## 6.2 The Framework of the iso-potential lines approach

Using the concepts of the preceding section, we are now able to present the iso-potential lines approach to path planning.

### Framework of iso-potential lines approach to path planning

**Step 1 (Initial constraint setup)** Determine initial hard constraints (ridges, feature lines, collision). Generate a covering-generation control graph of potential sources and clipping borders based on the hard constraints. Adapt the mesh resolution to the control lines.

**Step 2 (Milling error)** Adapt the potential dependent on the local milling error by potential scaling.

#### Step 3 (Initial potential and isolines)

- 3.1 Calculate the potential field;
- 3.2 Calculate isolines of the resulting potential field;

#### Step 4 (Potential and isoline optimization)

- 4.1 Minimize the contour graph.
- 4.2 Smooth the contour lines which are continuous but not necessarily smooth, by adaptation of the potential.
- 4.3 Introduce soft constraints in order to optimize the fairness of the resulting isolines with respect to tool acceleration and recalculate the isolines.
- 4.4 Iterate step 4.

**Step 5 (Concatentation of isolines)** Concatenate the elements of a maximal stack of related iso-lines into one spiral or zig-zag-curve. Smooth the spiral.

**Step 6 (Arrangement of curve segments)** Arrange the curve segments of the preceding in an order which minimizes contact-free tool motion.

**Step 7 (Freespace optimization and tool path extraction)** Optimize the assignment of freespace components and extract a concrete path of tool configurations from the freespace components.

The potential lines approach is based on continuous concepts, and it will be explained in the following in this way.

One approach to its implementation is discretization. Discretization is achieved by approximating the workpiece surface by a triangular mesh. The effect of discretization artifacts depends on the coarseness of the mesh. One possibility to cope with them is to perform a simulation with the resulting paths. If errors occur, the mesh may be refined in order to start a new trial. In the following, algorithms based on this approach to discretization are presented. They concern all steps of the algorithm except of step 7 which is performed according to section 4.4.

### 6.3 Covering-generating control graph

The covering-generating control graph is subject of the steps 1 and 4 of the algorithm framework of section 6.2.

Types of control lines are collision control lines, accessibility control lines, the boundary lines of  $S^I$ , feature lines, tool movement control lines, and contour-graph-reducing control lines, assuming that the region of interest is defined so that no collision-prevented accessibility problem exists. The first two types of control lines define hard constraints, the third one is between hard and soft, and the last two are soft constraints. Soft constraints mean that solutions not taking them into account work, but they are possible improvable, while hard constraints have to be satisfied.

#### 6.3.1 Collision Control Lines

A singular point (cf. section 4.3.1) is called *regularly millable* if it is within the tolerance radius of regular point. If all singular points are well millable, paths

not containing singular points are sufficient for milling the whole surface of interest. A finite set of regions, each bounded by finitely many closed curves without singular points, is called a *millable approximation* of the singular points, if every singular point is in one of the regions and within the tolerance radius of a point of the boundary curve. The boundary curves of a millable approximation can be used as collision control lines.

A *partially millable* approximation of the singular points is a finite set of regions which contain all singular points, but not all of those are in the tolerance radius of a boundary point. The regions containing such singular points will be treated in a possibly inefficient way by a path-based treatment of collisions.

### **Discrete implementation**

We use the discrete implementation of collision analysis on curves of section 4.3.2 for constructing collision control lines. We consider the set of all triangles incident to singular edges (cf. section 4.3.2). We call those triangles *singular*. The region covered by the singular triangles define a partially millable approximation. The boundary edges define the collision control lines.

### **6.3.2 Accessibility Control Lines**

The accessibility control lines are defined by the border between the set of accessible and non-accessible workpiece points.

#### **Discrete implementation**

The existence of accessibility is determined for every vertex. The non-accessible vertices are removed from the mesh. The possibly resulting mesh boundary is the accessibility constraint.

In order to achieve a good approximation of the accessibility constraint, the mesh can be refined in the environment of the border.

### 6.3.3 Boundary lines

The boundary lines of  $S^I$  can possibly be chosen in a way that supports favorable isolines. The only constraint is that  $S^I$  can still be produced. The goal is to optimize the fairness of tool acceleration.

#### Discrete implementation

The boundary lines are given by closed paths on the graph defined by the workpiece mesh.

### 6.3.4 Feature lines

Feature lines are lines which should not be traversed by a tool path. Examples are sharp edges or ridges where the workpiece surface has infinite curvature, i.e. not differentiable, or at least a high absolute curvature in one direction.

Feature lines are possibly explicitly milled in an additional post-processing step.

#### Discrete implementation

The feature lines are given by a subgraph of the graph defined by the workpiece mesh.

### 6.3.5 Tool movement control lines

Tool movement control lines may be introduced in order to force the flow of the isolines in preferred directions. From the view of the production machine, rapid changes of acceleration of machine parts or the tool should be avoided. Acceleration may have two reasons: changing the amount of the speed and changing the direction of the speed. The latter has geometric aspects.

Concerning normal curvature and geodesic curvature, the normal curvature is influenced by the shape of the surface, while the geodesic curvature depends on the constraints caused by surface boundaries and collisions.

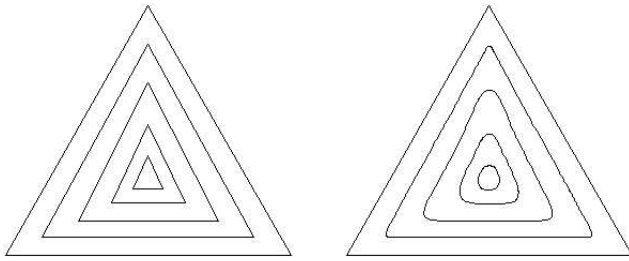


Figure 18: Sharp corners (left) are removed by curve smoothing.

There is a conflict between the amount of tool acceleration and the milling error. From the view of the milling error, the tool should move perpendicularly to the lines of low curvature because the approximation of the tool silhouette and the workpiece silhouette with the normal plane orthogonal to the direction of movement should be better than in the opposite case. On the other hand, this means tool movement along the line of maximum curvature which may imply higher acceleration.

Regions with rapidly changing maximum or minimum normal curvature are of interest since the respective normal curve could have rapidly changing curvature there. If just one of the both curvature functions has this property, then the curves of the other should be preferred.

Regions of highly different maximum and minimum normal curvature are of interest since the normal curvature of a curve may change there rapidly.

Regions of high local geodesic curvature on an potential source may cause a high change of curvature.

In conclusion from this discussion, the following modifications may improve paths:

1. Control lines may be inserted along a fair minimum/maximum normal curvature line if the intersection maximum/minimum curvature lines have rapidly changing curvature. Locations of this type are also candidates for feature lines.
2. Control lines may be inserted in order to cut corners of existing potential

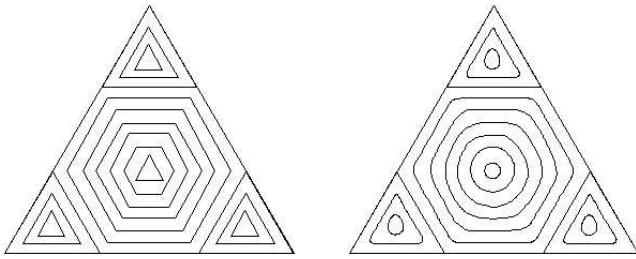


Figure 19: Fairing curves by corner cutting.

sources. This may cause new regions with unfavorable behavior of the isolines, but of possibly small size (figure 19).

A difficulty of corner cutting could be that lift-offs of the tool are necessary which may cause production artifacts at the borders between segments.

### 6.3.6 Contour-graph-reducing control lines

The vertices of a *contour graph* are pairs  $(\mathbf{p}, c)$  of a surface point and a potential value  $c$  so that a contour vanishes, or contour points touch at  $\mathbf{p}$  when  $c$  is reached. Figure 20) shows isocontours of a polygon where all edges are potential sources. One stack of contours belonging to lower potential values is vanished. Two vertices  $(\mathbf{p}_1, c_1)$  and  $(\mathbf{p}_2, c_2)$  are connected by an edge if a curve of monotonous potential values exists between  $\mathbf{p}_1$  and  $\mathbf{p}_2$  so that the contours containing any point  $(\mathbf{p}, c)$  on the curve do not meet any further vertex. Every edge corresponds to a region over which the isolines form a stack of co-centric contours. Figure 21, top, shows the contour graph of sample polygon. The concept of contour graphs is similar to the concept of Reeb graphs in the application of Shinagawa et al. [Shi91].

A different view on the contour graph is the skeleton (figure 21, bottom). The *skeleton* consists of all points which have equal distance to at least two points on potential sources. The vertices of the contour graph are the points with locally maximal potential value and the interior points with local minimum on the skeleton. The edges correspond to the lines of the skeleton connecting two

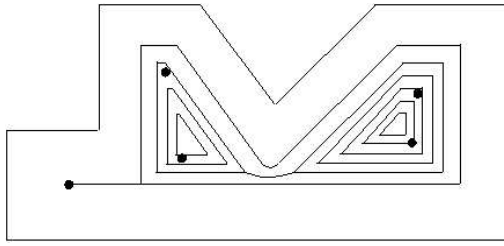


Figure 20: Isocontours of a polygon where all edges are potential sources. One stack of contours belonging to lower potential values is vanished.

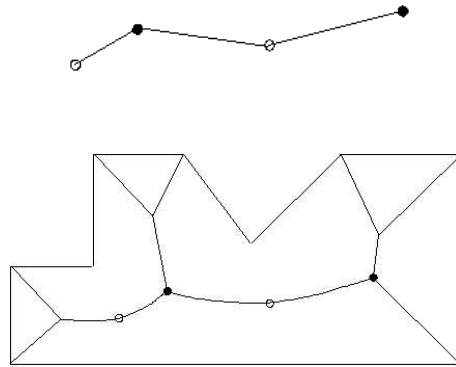


Figure 21: The contour graph (top) and the skeleton (bottom) of a polygon.

consecutive extremal points.

For path planning, the number of stacks of co-centric contours should be minimized since each of them implies a lift-off of the tool. Further, a high number of contour stacks may lead to many of them with just small extension which should be avoided.

Since every edge of a contour graph corresponds to a stack of co-centric contours, the number of edges should be reduced. An approach to achieving this goal is decomposing a segment into sub-segments of simpler shape. For example, consider a non-convex polygonal region in the plane and its decomposition

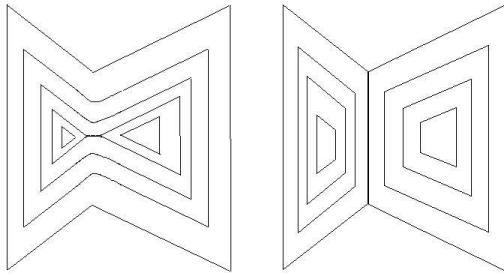


Figure 22: Reduction of the number of contour stacks by insertion of a chord. The left image shows three contour stacks which are replaced by two contour stacks in the right image.

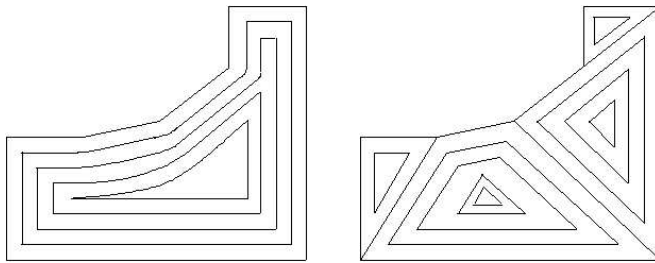


Figure 23: Convex decomposition does not always reduce the number of contour stacks. However, the shapes of the isocontours in the right image do not show sharp bendings as the curves in the right image.

into convex subregions by chords. The isolines of a convex polygon form a co-centric stack so that the given region is now covered by a number of stacks equal to the number of convex subpolygons. As the example in figure 22 shows this is better than the original result. However, as figure 23 shows, convex decomposition does not always improve the number of co-centric stacks, but sharp bendings are reduced.

Based on those observations we propose the following heuristics of decomposition:

1. Calculate the local minimal vertices of the contour graph.
2. For every local minimal vertex:
  - 2.1 Calculate the at least two points on the potential sources which are closest to it.
  - 2.2 For all pairs of those points:
    - If the shortest curve between the pair of points is completely in the region:
      - 2.2.1 Insert the curve as an potential source into the control graph
      - 2.2.2 Recalculate the iso-lines and save the result if it is better than the one obtained up to now.
      - 2.2.3 Remove the inserted curve.

The test for quality in step 2.2.2 may concern the number of co-centric stacks, but additional criteria like e.g. the quality of the shape of the isolines may become involved.

Later-on, further heuristics for reducing the number of contour graph edges will be proposed.

### **Discrete implementation**

The vertices of the contour graph for the mesh approximation of the workpiece are located at mesh vertices. The reason is that the marching triangle algorithm constructs just single curve traversals of the interior of a triangle. If all incident vertices of mesh vertex have smaller or equal potential values, the mesh vertex is a contour graph vertex representing a local maximum. If the incident vertices of a mesh vertex  $\mathbf{p}$  are traversed in a clockwise way, and at least two changes from a smaller to a higher potential value than that of  $\mathbf{p}$  occur, then  $\mathbf{p}$  is a local minimum vertex of the contour graph. The edges incident to a contour graph vertex  $(\mathbf{p}, c)$  are found by sweeping the triangles of a mesh starting with those incident to  $\mathbf{p}$  in order to successively visit further mesh vertices  $\mathbf{q}$  in falling potential value order, starting at  $c$ . If an isocontour traversing  $\mathbf{q}$  contains a contour graph vertex  $\mathbf{r}$ , a directed edge from  $\mathbf{p}$  to  $\mathbf{r}$  is inserted in the contour graph. This procedure is performed for every contour graph vertex. Finally, transitive edges are removed.

The constructed vertices may be cross-checked with closest potential source vertices stored at every vertex of the mesh. The contour graph vertices should be just those which have more than one closest vertex. However, discretization artifacts could induce deviations. An open question is to check for whether such an inconsistency may happen.

### 6.3.7 Combination of the lines into a control graph

The control graph is obtained by taking the union of all control lines involved. The curve segments are defined by equivalence classes on the points of the control lines, except those which consist of a single point which are the vertices. Two points are equivalent if they are on exactly the same control lines.

### Discrete implementation

The control lines are represented by subgraphs of the workpiece mesh. Their union is the control graph.

## 6.4 Adaptation of contour lines to the accessibility error

Adaptation of contour lines to the accessibility error is subject of step 2 of the algorithm framework of section 6.2. In the following, it is achieved by suitable distance scaling.

In the accessibility analysis of section 4.1 we have defined, among other data, the radius of influence  $r(\mathbf{p}, \mathbf{t}, \mathbf{R})$  of every accessible point  $\mathbf{p} \in S^I$ . If  $\mathbf{p}$  is regular (cf. section 4.3.1), we define  $r(\mathbf{p})$  as the average over  $r(\mathbf{p}, \mathbf{t}, \mathbf{R})$  in the freespace component of  $\mathbf{p}$ . If  $\mathbf{p}$  is singular, we set  $r(\mathbf{p}, \mathbf{t}, \mathbf{R}) := 0$

If the scaling value of  $\mathbf{p}$  is set on  $s(\mathbf{p}) := 1/(\alpha \cdot r(\mathbf{p}))$ ,  $1 \leq \alpha \leq 2$ , then the slope of the potential function is adapted so that the distance of isolines for consecutive integer values  $c$  and  $c + 1$  is about  $r(\mathbf{p})$ .  $\alpha$  controls the degree of overlap of neighboring tool traces. Since the potential function is defined for regular points only in our application,  $r(\mathbf{p})=0$  is excluded. In order to be on the safe side with the demanded error, the slopes could possibly be slightly increased.

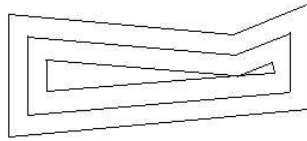


Figure 24: Occurrence of a small contour stack.

A less sophisticated solution is taking a global constant slope which is chosen according to the smallest distance between isolines necessary to achieve the desired error bound.

### Discrete implementation

The scaling values are assigned to the edges of the workpiece mesh. They are calculated from the  $r$ -values stored at the mesh vertices. We propose to take the inverse of the average of the  $r$ -values of the two end vertices of the edge. Alternatively, the further vertices of the triangles incident to the edge may be taken into account. However, if the mesh is highly curved at the edge and the edge is assumed to be oriented in direction of the lowest curvature, the involvement of just the edge vertices should be preferred.

## 6.5 Contour-graph reduction

The goal here is to remove small co-centric contour stacks (Figure 24). The approach can be applied at local minimum points of the skeleton.

Let  $(\mathbf{p}, c(\mathbf{p}))$  be such a local minimum point, and  $(\mathbf{q}, c(\mathbf{q}))$  be an adjacent vertex with smallest potential value (may be a local maximum or local minimum). We consider all isolines for values  $c$ ,  $c(\mathbf{p}) < c < c(\mathbf{q})$ , intersecting the skeleton incident to  $\mathbf{p}$ . Every isoline  $\mathbf{k}[c]$  with isovalue  $c$  intersects the skeleton at least once. The intersection points subdivide  $\mathbf{k}[c]$  in curve segments. The end points of the curve get the local maximum value  $c(\mathbf{q})$  as new potential value. The curve point of equal distance from the endpoints is set on  $c$ . The points in-between are interpolated continuously and monotonously between those three points relative to a curve length parametrization, e.g. linearly in the simplest

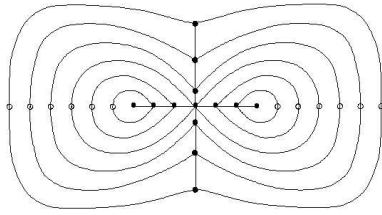


Figure 25: Explanation of the procedure of removal of a contour stack.

case.

Figure 25 illustrates the procedure. Let  $c_{\min}$  be a value less than  $c(\mathbf{p})$  so that no contour graph vertex occurs on the iso-contours of values in-between which should be as small as possible. At least two points exist on the isoline of  $c_{\min}$  which have the same minimum distance to  $\mathbf{p}$ . The connecting curves intersect the isolines in the interval between  $c_{\min}$  and  $c(\mathbf{p})$ . These intersections split the iso-lines in curve segments. The curve segments are treated like in the preceding paragraph. The only difference is that the potential value at the intersections is chosen as an interpolated value between  $c_{\min}$  and  $c(\mathbf{q})$ , according to some monotonous interpolation along the connecting curves, e.g. linear interpolation. In this way a new potential function is achieved which eliminates the local minimum point.

### Discrete implementation

The potential intervals  $(c(\mathbf{p}), c(\mathbf{q}))$  and  $(c_{\min}, c(\mathbf{p}))$  are swept by calculating the involved isocontours traversing mesh vertices. For every mesh vertex on these isocontours, its new value is calculated by interpolation on the isocontour. The connecting curves required for the second interval are calculated by intersection of the shortest paths between  $\mathbf{p}$  and potential source vertices stored at  $\mathbf{p}$  with the isocontour of  $c_{\min}$ .

## 6.6 Potential and Isoline Smoothing

The boundary of a region can be smoothed by removing in a first step all points of the region which are closer than a given distance  $\delta > 0$  to a boundary point. In a second step, all points outside the region which have at most distance  $\delta$  to the boundary of the new region are added to the new region. In the terminology of computational morphology, the first operation is an example of an erosion, while the second operation is a dilatation [Gon02]. This approach smoothes convex segments of the boundary. If the same procedure is applied to the complement region, the concave segments are smoothed, too. Since in our application, concave segments are smoothed automatically, smoothing of convex segment suffices.

The morphological smoothing can be applied to the individual isolines and to the whole potential function. The latter is probably more efficient. In the first case a spherical structuring element can be taken. A sphere of radius  $l$  with center  $\mathbf{p} \in S^I$  over the surface  $S^I$  is defined relatively to the surface metrics  $d$  by the functions  $h(\mathbf{q}) := \pm\sqrt{l^2 - d(\mathbf{q}, \mathbf{p})}$ ,  $\mathbf{q} \in S^I$ . The radius of the structuring element can possibly be made dependent on the potential value of  $\mathbf{p}$ , e.g. linearly in the value with a factor between 0 and 1. ...

## 6.7 Concatenation of Isocurves and Arrangement of Curve Segments

The concatenation of isocurves into longer contact curve segments and the arrangement of curve segments in an order which minimizes the contact-free tool movements are subject of the steps 5 and 6 of the algorithm. They can be performed as follows.

### 6.7.1 Concatenation of Isocurves

We have two types of isocurves: open curves which are adjacent to a clipping line, and closed curves which are not. Both types are treated separately. The closed isoline are concatenated into spirals, while the open isolines become zig-zag curves (figure 26).

The set of all closed isocurves is divided into disjoint stacks of co-centric isocurves by a relation of equivalence. Two isocurves are *directly equivalent* if

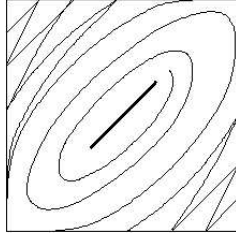


Figure 26: Concatenation of isocurves in to spiral and ziz-zag curves segments.

they belong to neighboring isovalues and if one of them is inside the other one, being the single one with this property and the same isovalue. Two isocurves are *equivalent*, if a sequence of directly equivalent pairs of isolines exist so that one belongs to the first pair and one belongs to the second pair. Further, an isoline is equivalent to itself. The classes of equivalence of the relation are the desired stacks of co-centric isocurves.

The set of all open isocurves is divided into disjoint stacks by considering two isocurves as *directly equivalent* if they belong to neighboring isovalues and they are neighboring on a clipping line at both ends. Neighboring means that the endpoints are connected by a segment of a clipping curve, and that the two isolines are locally on the same side of the clipping curve. Direct equivalence is extended to an equivalence relation like for closed isocurves. The resulting equivalence classes are the stacks of open isocurves.

The elements of a stack of closed isocurves are replaced by spiral segments which together induce one spiral curve. One approach to replacing by spiral segments is starting at a point  $\mathbf{p}_0$  of the largest isocurves  $\mathbf{k}_0$ . Then a point  $\mathbf{p}_1$  closest to  $\mathbf{p}_0$  on the next inner ring  $\mathbf{k}_1$  is selected, and so on. A curve length parametrization scaled onto  $[0, 1]$ , starting an ending at  $\mathbf{p}_i$  is performed on every ring  $\mathbf{k}_i$ . The sprial segment between  $\mathbf{k}_i$  and  $\mathbf{k}_{i+1}$  consists of the points  $\bar{\mathbf{k}}_i(t)$ ,  $t \in [0, 1]$ , which subdivide the shortest path from  $\mathbf{k}_i(t)$  and  $\mathbf{k}_{i+1}(t)$  in the ratio  $t : (1 - t)$ . The first and the last ring may be added to the set of resulting spiral segments.

The elements of a stack of open isocurves are concatenated into a zigzag curve by connecting the end points alternatingly by fading segments, like shown in figure 27. The fading curves can be constructed analogously to the spiral seg-

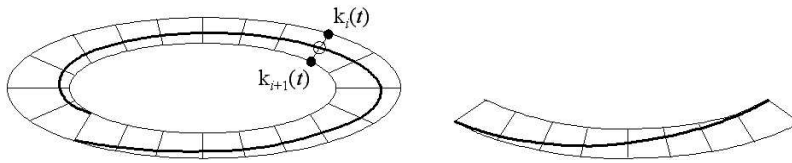


Figure 27: Construction of a spiral (left) and a zigzag (right) segment from two subsequent isocurves.

ments, see figure 27.

### 6.7.2 Arrangement of curve segments

The constructed curve segments have to be arranged in some linear order within the final tool movement path. This problem can be understood as traveling salesman problem with the constraint that a number of given edges have to be part of the roundtrip. The vertices of the underlying graph  $G$  are the end points of the curve segments. An edge is introduced for every pair of vertices. The edges which have to be part of the roundtrip are those between the two end points of a given curve segment. These edges get the weight 0, while the other edges are weighted by the costs of a tool movement in the environment of the workpiece from one end to the other, see section 5.3.4.

If the number of curve segments is small, the number of vertices in the described traveling salesman graph is small, and an optimum solution can be calculated by systematic enumeration. Systematic enumeration can for example be performed by constructing the following type of bi-partite subgraphs of the graph  $G$ . The set of vertices of  $G$  is divided in two equal sized subsets  $V_1$  and  $V_2$ , so that the two end points of every curve segment are in different subsets. All edges of  $G$  with both end points just in  $V_1$  or just in  $V_2$ , as well as the edges corresponding to the given curve segments, are removed. In the resulting bi-partite graphs a complete matching of minimum weight is determined, and the minimum over all of them is reported as solution, i.e. the end points of the environment paths connecting the given curve segments.

A matching in a graph is a set of edges which do not share any end point. A complete matching is a matching for which every vertex of the graph is an end

point of one of its edges. The weight of a matching is the sum of the weights of its edges. It can be easily seen that the constructed bi-partite graphs have a complete matching.

Let  $q$  be the number of path segments. Then the number of bi-partite subgraphs to be investigated is  $2^q$ . For every bi-partite subgraph,  $q!$  possible complete matchings exist. This means that by enumerative search  $2^q \cdot q!$  sums have to be evaluated. Heuristics may help to reduce the number, in particular if an approximate solution is sufficient. For example, in a first step, a bi-partite graph of minimum edge weight may be determined, in which in a second step a best matching is found. This reduces the number of steps of a brute force enumeration to  $2^q + q!$ . For both steps, non-enumerative heuristics, for example greedy approaches and pruning, may be used to further restricting the search spaces.

## 7 Conclusion

Two approaches to automatic path planning have been presented. They are based on a discrete approximation of the continuous problems which is achieved by meshing of the workpiece surface and regular sampling of the tool configurations. Both approaches have been developed for complex planning tasks for which the significant computational requirement possibly not necessary for simple situations is justified.

Future work may concern the combination with simpler and more efficient approaches adequate for simple planning tasks, like for surfaces of little curvature without expected severe collision problems, as a step towards the application in practice. One way of achieving this is its integration into an existing semi-automatic path planning or path modeling tool, as an additional component to be applied on parts of a workpiece for which path planning is difficult with existing solutions.

A particular interesting aspect with respect to interaction is semi-automatic optimization by interactive steering of the behavior of the automatic algorithm. The iso-potential line approach seems to be particularly suitable for this purpose because of its intuitive control parameters given by the control lines and the control graph.

Another issue is the extension of the two approaches to other production technologies. Kim and Sarma [Kim03] include other path-oriented production tech-

nologies into their model. Examples are painting and coating of surfaces by spraying using industrial robots. We have started work on robot-based thermal spraying and it seems that the concepts can be transferred to this and other problems analogously to that of [Kim03].

Besides the production technology, surface covering have applications for example in applications of autonomous robots [Cho01]. An example is cleaning of floors by an autonomous robot which can also be understood as a traveling salesman problem with constraints on the curvature of the paths.

## References

- [Ark01] E.M. Arkin, M.A. Bender, E.D. Demaine, S.P. Fekete, J.S.B. Mitchell, S. Sethi, Optimal covering tours with turn costs, Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 01), 2001, 138–147
- [Ayasse02] J. Ayasse, H. Müller, Sculpturing on discrete displacement fields, In: Proc. Eurographics 2002, Computer Graphics Forum 21(3), 2002, 431–440
- [Bae00] T. Bäck, D. B. Fogel, T. Michalewicz, Evolutionary computation 1: basic algorithms and operators, IoP, 2000
- [Bal02] M. Balasubramaniam, P. Laxmiprasad, S. Sarma, Z. Shaikl, Generating 5-axis NC roughing paths directly from tessellated representation, Computer Aided Design 32(4), 2000, 261–277
- [Bla73] W. Blaschke, K. Leichtweiss, Elementare Differentialgeometrie, Springer-Verlag, Berlin, 1973
- [Bot82] M. Botsch, L. Kobbelt, Resampling feature and blend regions in polygon meshes for surface reconstruction, In: Proc. Eurographics 2001, Computer Graphics Forum 20(3), 2001, C-402–C-410
- [Car76] M.P. Do Carmo, Differential Geometry of Curves and Surfaces, Prentice Hall, Englewood Cliffs, NJ, 1976.
- [Chi02] I.J. Chiou, Y.S. Lee, A machining potential field approach to tool path generation for multi-axis sculptured surface machining, CAD 34, 2002, 357–371

- [Cho99] B.K. Choi, R.B. Jerard, *Sculptured Surface Machining: Theory and Applications*, Kluwer Academic Pub., 1999
- [Cho01] H. Choset, Coverage for robotics – A survey on recent results, *Annals of Mathematics and Artificial Intelligence* 31, 2001, 113–126
- [Cli88] H.E. Cline, W.E. Lorensen, S. Ludke, C.R. Crawford, B.C. Teeter, Two algorithms for the three-dimensional Construction of Tomograms, *Medical Physics* 15(3), 1988, 320–327
- [Din03] S. Ding, M.A. Mannan, A.N. Poo, D.C.H. Yang, Z. Han, Adaptive iso-planar tool path generation for machining of free-form surfaces, *CAD* 35, 2003, 141–153
- [Dra97] D. Dragomatz, S. Mann, A classified bibliography of literature on NC milling path generation, *CAD* 29, 1997, 239–247
- [Elb94] G. Elber, E. Cohen, Accessibility in 5-axis milling environment, *CAD* 26, 1994, 490–496
- [Elb97] G. Elber, R. Fish, 5-axis freeform surface milling using piecewise ruled surface approximation, *ASME Journal of Manufacturing Science and Engineering* 119, 1997, 383–387
- [Flu02] A. Flutter, J. Todd, A machining strategy for toolmaking, *CAD* 33, 2001, 1009–1022
- [Fri00] S.F. Frisken, R.N. Perry, A.P. Rockwood, T.R. Jones, Adaptively sampled distance fields: A general representation of shape for computer graphics, In: *Proc. SIGGRAPH 2000*, pp. 249–254, 2000
- [Gar79] M.R. Garey, D.S. Johnson, *Computers and intractability – a guide to the theory of NP-completeness*, W.H. Freeman and Comp., New York, 1979
- [Gla99] G. Glaeser, J. Wallner, H. Pottmann, Collision-free 3-axis milling and selection of cutting tools, *CAD* 31, 1999, 225–232
- [Gre85] J. Grefenstette, R. Gopal, B. Rosmaita, D. Van Gucht, Genetic algorithms for the traveling salesman problem, In: *Proc. of the International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, PA, 1985, 160–168

- [Gon02] R. Gonzalez, R. Woods, Digital Image Processing (2nd ed.), Prentice-Hall, 2002
- [Gol85] D. E. Goldberg, R. Lingle, Alleles, loci, and the traveling salesman problem, In: Proc. of the International Conference on Genetic Algorithms and Their Applications, Pittsburgh, PA, 1985, 154–159
- [Got96] S. Gottschalk, M.C. Lin, D. Manocha, OBBTree: A hierarchical structure for rapid interference detection, In: Proc. SIGGRAPH 1996, pp. 171–180, 1996
- [Gra02] P. Gray, S. Bedi F. Ismail, Rolling ball method for 5-axis surface machining, CAD, to appear
- [Had06] Ch. Hadac, Generierung optimierter Oberflächennetze für die 3D-FE-Simulation auf Basis von CT-Daten, Diplomarbeit, Informatik VII, Universität Dortmund, 2006
- [Has04] D. Hasanbegovic, Bahnplanung für fünfachsige Fräsen unter besonderer Berücksichtigung von Werkzeug-Werkstück-Kontakten, Diplomarbeit, Informatik VII, Universität Dortmund, 2004
- [Hel91] M. Held, On the computational geometry of pocket machining, Lecture Notes in Computer Science 500, Springer-Verlag, 1991
- [Hel00] K. Helsgaun, An effective implementation of the Lin-Kernighan traveling salesman heuristic, European J. Oper. Res. 126(1), 2000, 106–130
- [Hel02] K. Helsgaun, User Guide for the LKH Algorithm, Juli 2002, [http://homer.ruc.dk/~keld/public\\_html/research/LKH](http://homer.ruc.dk/~keld/public_html/research/LKH) (März 2004).
- [Ho01] S. Ho, S. Sarma, Y. Adachi, Real-time interference analysis between a tool and an environment, CAD 33, 2001, 935–947
- [Hos92] J. Hoschek, D. Lasser, Grundlagen der geometrischen Datenverarbeitung, 2. Aufl., B.G. Teubner, Stuttgart, 1992
- [Hua94] Y. Huang, J.H. Oliver, NC milling error assessment and tool path correction, Proc. SIGGRAPH 1994, 287–294

- [Ima86] H. Imai, M. Iri, Computational-geometric methods for polygonal approximation of a curve, *Computer Vision, Graphics and Image Processing* 36, 1986, 31–41
- [Jen02] C.G. Jensen, W.E. Red, J. Pi, Tool selection for five-axis curvature matched machining, *CAD* 34, 2002, 251–266
- [Jer89] R.B. Jerard, S.Z. Hussaini, R.I. Drysdale, B. Schaudt, Approximate methods for simulation and verification of numerically controlled machining programs, *The Visual Computer* 4, 1989, 329–348
- [Jue97] M. Jünger, G. Reinelt, G. Rinaldi, The traveling salesman problem, In: M. Dell’Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, John Wiley & Sons Ltd., Chichester, UK, 1997, 199–221
- [Kim02] T. Kim, S.E. Sarma, Toolpath generation along directions of maximum kinematic performance: a first cut at machine-optimal paths, *CAD* 34, 2002, 453–468
- [Kim03] T. Kim, S.E. Sarma, Optimal Sweeping Paths on a 2-Manifold: A New Class of Optimization Problems Defined by Path Structures, *IEEE Transactions on Robotics and Automation* 19(4), 2003, 613–636
- [Kob00] L. Kobbelt, T. Bareuther, H.-P. Seidel, Multiresolution shape deformations for meshes with dynamic vertex connectivity, *Computer Graphics Forum* 19(3), 2000, C-249–C-259
- [Kri96] V. Krishnamurthy, M. Levoy, Fitting smooth surfaces to dense polygon meshes, *Proc. SIGGRAPH 1996*, 313–324
- [Lat91] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991
- [Lau02] B. Lauwers, P. Dejonghe, J.P. Kruth, Optimal and collision free tool posture in five-axis machining through the tight integration of tool path generation and machine simulation, *CAD*, to appear
- [Law85] E. Lawler, J. Lenstra, A. Rinnooy Kan, D. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, 1985

- [Lee97] Y.S. Lee, Admissible tool orientation control of gouging avoidance for 5-axis complex surface machining, *CAD* 29, 1997, 507–521
- [Lee98] Y.S. Lee, Non-isoparametric tool path planning by machining strip evaluation for 5-axis sculptured surface machining, *CAD* 30, 1998, 559–570
- [Len99] C. Lennerz, E. Schömer, T. Warken, A framework for collision detection and response, In Proc. 11th European Simulation Symposium (ESS'99), 309–314
- [Li94] S.X. Li, R.B. Jerard, 5-axis machining of sculptured surfaces with a flat-end cutter, *CAD* 26, 1994, 165–178
- [Lin73] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, *Operations Research* 21, 1973, 498–516
- [Mar94] S. Marshall, J.G. Griffiths, A survey of cutter path construction techniques for milling machines, *International Journal of Production Research* 32, 1994, 2861–2877
- [Mey99] H.W. Meyer, F. Albersmann, Wochenpensum in zwei Stunden: Features und NC-Sets verkürzen die NC-Programmierung im Formenbau, *Special Tooling* 5, 1999, 42–48
- [Mic92] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1992
- [Oli90] J.H. Olivier, E.D. Goodman, Direct dimensional NC verification, *Computer Aided Design* 22, 1990, 3–10
- [Par02] S.C. Park, Y.C. Chung, Offset tool-path linking for pocket machining, *CAD* 34, 2002, 299–308
- [Per01] R.N. Perry, S.F. Frisken, Kizamu: A system for sculpting digital characters, In: Proc. SIGGRAPH 2001, pp. 47–56, 2001
- [Pot99] H. Pottmann, J. Wallner, G. Glaeser, G. Ravani, Geometric criteria for gouge-free three-axis milling of sculptured surfaces, *ASME J. of Mechanical Design* 121, 1999, 241–248

- [Roe00] C. Rössl, L. Kobbelt, H.-P. Seidel, Line art rendering of triangulated surfaces using discrete lines of curvature, in: Proc. 8th International Conference in Central Europe on Computer Graphics (WSCG'00), 2000, 168–175
- [Rup98] D. Ruprecht, H. Müller, A scheme for edge-based adaptive tetrahedron subdivision, In: Mathematical Visualization (H.-C. Hege, K. Polthier, eds.), pp. 61–70, Springer-Verlag, 1998
- [Sch92] W.J. Schroeder, J.A. Zarge, W.E. Lorensen, Decimation of triangle meshes, In: Proc SIGGRAPH 1992, pp. 65–70, 1992
- [Shi91] Y. Shinagawa, T.L. Kunii, The Homotopy Model: A Generalized Model for Smooth Surface Generation from Cross Sectional Data, The Visual Computer 7, 1991, 72–86
- [Ste86] R.E. Steuer, Multiple Criteria Optimization: Theory, Computation and Application, Wiley, 1986
- [Sto05] M. Stöneberg, Minimierung von Umkehrpunkten bei der Erzeugung günstiger Bahnen für das fünfachsige Fräsen, Diplomarbeit, Informatik VII, Universität Dortmund, 2005
- [Sto99] A. Storr, H. Ströhle, Rechnerunterstützte Produktion mit fünfachsigen spanenden Bearbeitungseinrichtungen, wt Werkstattstechnik 89, 1999, 78–82
- [Tan98] J.W.H. Tangelder, J.S.M. Vergeest, M. Overmars, Interference-free NC machining using spatial planning and Minkowski operations, CAD 30, 2000, 277–286
- [Tur92] G. Turk, Re-Tiling polygonal surfaces, In: Proc SIGGRAPH 1992, pp. 55–64, 1992
- [Wei01] K. Weinert, T. Surmann, Approaches for Modeling Engagement Conditions in Milling Simulations, in Proc. 4th CIRP International Workshop on Modelling of Machining Operations, C.A. van Lutterfeld (ed.), Morgan-Kaufman Publ., 2001, 67–70
- [Wei02] K. Weinert, H. Müller, W. Kreis, T. Surmann, J. Ayasse, T. Schübstuhl, K. Kneupner, Diskrete Werkstückmodellierung, ZWF 97, 2002, 385–389

- [Yan99] D.C.H. Yang, Z. Han, Interference detection and optimal tool selection in 3-axis NC machining of free-form surfaces, *CAD* 31, 1999, 303–315
- [Zit03] E. Zitzler, M. Laumanns, S. Bleuler, A tutorial on evolutionary multiobjective optimization, In: *Workshop on Multiple Objective Metaheuristics (MOMH 2002)*, Springer-Verlag, 2003