Geometric and Algorithmic Aspects of Automatic Path Planning with Relation to Spray Deposition Processes

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Alexander Kout

Dortmund

2015

Tag der mündlichen Prüfung: 14.07.2015

Dekan:	Prof. Dr.-Ing. Gernot Fink
Gutachter:	Prof. Dr. Heinrich Müller
	Prof. Dr. Christoph Buchheim

Abstract

This thesis contains novel contributions to automatic path planning for path-oriented production techniques. They are mainly motivated by path planning for spray deposition processes. In contrast to milling, these processes have found less interest in the basic research community in the past.

The first result of this thesis is a novel two step approach for automatic path planning to achieve a desired coating height on free-form surfaces. It can be characterized as a path-geometry-last approach for tool path planning, in contrast to the widespread path-geometry-first approach. The purpose of the first step is to optimize process-related properties in a path-free form, a so-called gun configuration cover. The transfer onto a path is worked out in the second step for paths based on offset curves represented by isolines of geodesic distance functions.

This type of curves leads to the second emphasis of the thesis, improvements of offset curves. The advantage of these curves is that they are in widespread use and that the representation as isolines is stable and flexible. In this thesis they are considered on surfaces represented by triangular meshes, taking into account the state-of-the-art of mesh processing. This thesis addresses two issues of offset curves.

The first issue concerns the local adaptation of the offset distances to the requirements of the production processes. For this purpose, a novel approach employing anisotropic distance functions on triangular meshes is presented. The resulting adaptation method is applied to simply derive milling paths with uniform cusp heights. A second application concerns the already mentioned second phase of the path-geometry-last approach and an error-adaptive path interval adaptation for spray coating.

The second issue concerns two major drawbacks of isolines, sharp corners and complex topologies. The approach is based on weighted distance functions on triangular meshes and employs a novel concept of weighted medial axes. An experimental evaluation shows the usefulness of the approach.

The third emphasis of the thesis concerns the manipulator. This leads to a path placement problem where a given tool path has to be placed relative to the manipulator so that it becomes executable in an optimized way. The problem is formally specified as an optimization problem. An extensive experimental comparative analysis of different optimization methods is performed.

Furthermore, different approaches of relaxation of the original problem are presented. The first approach is to allow a time-dependent motion of the workpiece, with a penalty term for the amount of motion. The second approach is to allow a deviation from the given tool path, with a penalty term for the amount of deviation.

The main advantage of the relaxed approaches is the simultaneous variation and optimization of the manipulator path, which allows to find a solution near the predefined tool path, even if there is no feasible solution for the tool path as it is. This is important, because it cannot be guaranteed in advance that a tool path has a feasible solution. Another crucial advantage of the relaxed approaches is their suitability for redundant manipulators and for redundant tool paths, i.e. not fully predefined tool paths.

Contents

1 Introduction

In this chapter, the scope and the results of the thesis are outlined. It starts with a brief informal introduction to the manufacturing processes which serve as use cases for the presented concepts. These are path-oriented, computer-controlled processes for which robot-based spray coating, but also NC milling, are the most important examples for the thesis. Then the task of automatic path planning for such processes is considered from a practical as well as from a conceptual view. This is followed by a survey of the contributions and the organization of the thesis. Finally, publications by the author, related to the thesis, are commented.

1.1 Path-oriented, Computer-controlled Manufacturing

Path-oriented, computer-controlled manufacturing systems work by moving a tool along a path in order to modify a workpiece. Examples of production technologies where path-oriented systems are widely used include milling, grinding, roller burnishing [76] and spraying. In this thesis, fundamental concepts are presented whose performance will mainly be evaluated for spray coating, but also for milling if relevant. For that reason, robot-based spray coating and NC milling are briefly outlined in the following.

1.1.1 Robot-based Spray Coating

Spray deposition processes are manufacturing processes which deposit material on a workpiece. The material is delivered by a spray gun moving relative to the workpiece surface. A special kind of spray deposition process is thermal spraying. During the *thermal spray process*, molten particles are accelerated in a spray gun towards a prepared surface where they impact with a high velocity, flatten, and form a thin layer on the substrate [39]. In *robot-based spray coating* the spray tool is usually attached to the arm of an industrial robot (Fig. 1.1). The robot accomplishes the required motion of the spray gun by executing a robot program.

The main goal is to achieve the desired coating height. Hence, the material distribution function of the spray process must be available to automatically plan the tool path. A potential difficulty arises if the material flow rate of the spray gun cannot be controlled, which is usually the case for thermal spray coating. This practically relevant constraint is assumed in this thesis.

1.1.2 NC Milling

Machining is a process in which a cutting tool is used to remove small chips of material from a workpiece. To perform this operation, a relative motion is required between the tool and the workpiece. This relative motion is achieved in most machining operations by means of a primary rotational motion and a secondary translational motion, called *feed*. The shape of the tool combined with these motions produces the shape of the desired workpiece surface.

In milling, a rotating cutter with multiple cutting edges (Fig. 1.2(a)) is moved slowly relative to the material of a given workpiece to generate a surface (Fig. 1.2(b)). Ideally, the

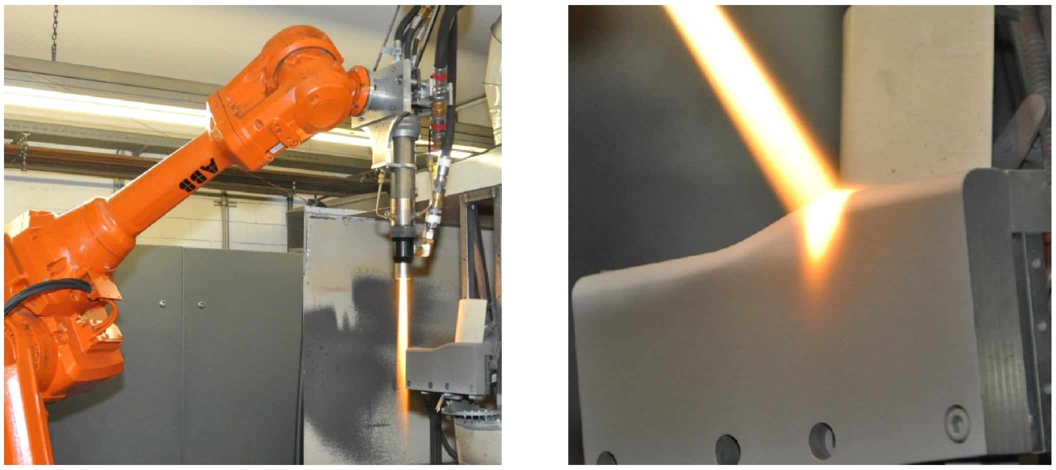

(a) (b)

Fig. 1.1: Thermal spray coating with an industrial robot.

direction of this feed motion is perpendicular to the tool's axis of rotation. This process is usually performed by specialized, so-called *NC milling machines* (Fig. 1.2(c)). *NC* stands for *numerical control* and means that the motion is performed based on a milling program which is executed by a control unit of the NC milling machine.

Milling operations can be divided in roughly two types: roughing cuts and finishing cuts. *Roughing cuts* are used to rapidly remove a large amount of material from the workpiece, i.e. with a large *material removal rate* (MRR), in order to produce a shape close to the desired form, but they leave some material on the workpiece for a subsequent finishing operation. *Finishing cuts* are used to complete the workpiece and achieve the final dimension, tolerances, and surface finish. Often, one or more roughing cuts are performed on the workpiece, followed by one or two finishing cuts.

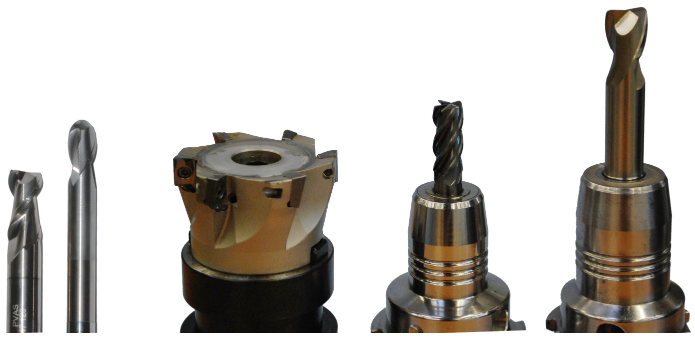
1.1.3 Comparison of Milling and Spraying

In the following, major differences between milling and spraying are outlined. For milling the discussion is restricted to the final finishing step.

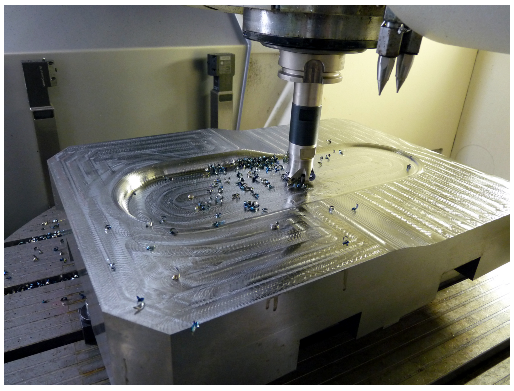
Spray coating differs from contact-oriented manufacturing methods like milling in that the applicator, that is the spray gun, is not in contact with the workpiece. This implies that the spray gun path may fill a larger volume of the space surrounding the workpiece. Furthermore, contact forces do not occur for spraying.

In milling, the contact forces limit the process time by their influence on the dynamics of the machine tool. A path that minimizes the dynamic load on the machine tool increases the feed rate and thus may reduce the process time.

In thermal spraying, the goal is not only to minimize the process time, which can be done by increasing the material flow rate, but also to be able to maintain the predefined velocity along the path, which is obviously difficult for high curvatures because of dynamic constraints of the robot. This velocity constraint is important to reach the desired coating height. Additionally, for thermal spray coating, the thermal variations in the workpiece should be low, so that the stress which is induced by the heat is minimized. This has the effect that the predefined velocity should be as high as possible.



(a)



(b)

(c)

Fig. 1.2: NC milling: (a) milling tools, (b) a milling process, and (c) the NC milling machine Deckel Maho DMU 50 eVolution.

In milling for finishing purposes, paths are often planned to be free of self-intersections. This may be reasonable for spray coating as well, but it can be abandoned if the velocity constraint cannot be fulfilled by this sort of paths.

Computer-based path planning for milling has found considerable interest in fundamental research and software development. In contrast, computer-based path planning for spray coating, and in particular thermal spray coating, is less well investigated which makes it an interesting field of research.

1.2 Automatic Path Planning from the Practical View

A central issue of path-oriented production processes is the generation of paths whose execution has the desired effects on the workpiece. Path planning on complex free-form surfaces and with manufacturing systems with many degrees of freedom, e.g. five-axis NC milling machines or industrial robots, is only little automated today. The reason is the complexity of the problem which is caused by the requirement to simultaneous consider the tool impact on the workpiece, the kinematics and the dynamics of the machine, and the collision avoidance of the system with the working environment. Especially in the context of machining, powerful interactive path planning systems exist, but the path planning is nevertheless still time-consuming. Other processes, like spray coating with industrial robots, are less examined. Extensions of path planning systems from machining exist but they usually do not take into account the particular properties of other processes, which may exclude favorable solutions. An aspect related to path planning is simulation. Simulation systems for the kinematics and dynamics of manipulators are in use, e.g. RobotStudio from ABB. Furthermore, simulation systems for specific manufacturing processes exist, in particular for machining. Those systems are mainly used for validation of paths but usually are not automatically integrated in the path planning process.

1.3 Automatic Path Planning from the Conceptual View

Path planning for manufacturing processes can be considered as an optimal control problem of a dynamic system. An optimal control problem consists of finding a time-dependent function which influences the behavior of a system, with the aim of optimizing the system behavior. Hence, an optimization has to be performed over the space of potential control functions. A difficulty of this type of problem are often high computational requirements.

The systems considered in the thesis contain

- a manipulator,
- a tool attached to the manipulator, and
- a workpiece influenced by the tool

as components. The goal is to achieve a desired final state of the workpiece, mainly expressed by a desired geometric shape, by using the manipulator to move the tool on an appropriate path. The central aspect of the thesis is to control the motion of the manipulator.

The set of system parameters which may be used as control parameters of the motion is not unique. A natural choice is to use the control parameters of the manipulator which are explicitly intended for that purpose, like e.g. angles or torques of the rotational joints of the robot arm. Those parameters influence the configuration of the manipulator and hence also the location of the tool attached to the manipulator. The motion of the manipulator induced by this kind of parameters will in the following be called *manipulator path*.

However, a further possibility is to use the motion parameters of the tool, e.g. the position and rotation of the tool in space. Those parameters induce a motion of the tool which will be called *tool path*. The tool path is usually expressed as the motion of a tool reference frame attached to the tool. Basically, a manipulator path may be constructed from the tool path by inverse kinematic calculation for the manipulator. However, this requires that the workpiece is appropriately placed relative to the manipulator. Finding a suitable workpiece placement in this context is known as *path placement problem*.

A further possibility is to consider the impact of the tool on the workpiece surface. An *impact path* is a path on the workpiece surface induced by the tool path. An example is the contact path of a milling tool on the desired goal surface (cf. Fig. 3.1(a)). An impact path may serve as a canonical basis of the parametrization of a tool path. In the milling example, a corresponding tool path may be defined by tool positions for which the contact path is tangential to the rotational tool hull. The resulting parametrization consists of the contact path and the tool orientation along the path.

These alternatives of control parameters lead to three approaches of optimization of the control function: *manipulator path optimization*, *tool path optimization*, and *impact path optimization*. An advantage of impact path optimization is that the impact paths are surface paths. As such, they may be chosen intuitively in a problem-adapted way, and their optimization may be computationally less costly. A disadvantage is that the properties of the tool and the manipulator are not taken into account. In not too complex environments, however, this approach may be useful since in this case path placement is not an issue.

The same advantage, but with less effect with respect to the problem complexity holds for tool path optimization. The simplification is that the properties of the manipulator are not considered. This may simplify path optimization in that it postpones manipulator aspects to the task of path placement. A disadvantage is that favorable solutions may be lost by not considering the manipulator.

Related work to these conceptual considerations will be presented in detail in those chapters of the thesis which emphasize the particular concept.

1.4 Contributions and Organization of the Thesis

The global aim of this thesis is to find methods for the as automated as possible generation of paths for manufacturing processes, where a tool with a bounded impact zone moves over a workpiece, such that it covers the surface and has the desired effect on the workpiece. The tool impact on the workpiece, which may contain free-form surfaces, as well as the machine kinematics are considered. Approaches related to impact paths and tool paths are presented, and the path placement problem is addressed. The particular problems tackled are mainly motivated by spray deposition processes. However, sometimes they are relevant for milling as well, and in those cases milling is considered, too.

The following sections give an overview of the major contributions of this thesis. Each of them corresponds to a chapter of the thesis. In a further, final chapter conclusions will be presented.

1.4.1 Tool Pose Optimization for Spray Deposition Processes

The aim of path planning for spray deposition processes is to find a time-dependent continuous sequence of spray gun configurations so that a coating of desired height is achieved when executing the sequence. Chapter 2 presents a novel approach to solve the planning task, called “path-geometry-last”, which leads to a more general gun configuration cover problem. The gun configuration cover problem is to find a finite set of spray gun configurations, which minimizes the error between a target coating and the coating induced by simultaneously activating those configurations. A suitable objective function for gun configuration covers is defined, and algorithmic solutions for the optimization problem are presented. An experimental evaluation shows that good approximations of the desired coatings can be achieved within reasonable computation times. In contrast to other approaches, the path-geometry-last variant gains an additional flexibility required to find application-oriented production paths for free-form workpieces.

1.4.2 Tool-adaptive Offset Paths on Triangular Mesh Workpiece Surfaces

In practice it is often preferred to use quasi-parallel offset curves as impact paths. Important examples are the contour-parallel and the direction-parallel curve patterns (Fig. 3.2). They require choosing the offset so that a tool moving along the curves has the desired impact at every surface point. A problem with the constant offsets commonly used is that in cases where the region of influence of a tool is different across the surface, an offset value necessary in one region may lead to a curve offset lower than required in other regions.

Chapter 3 introduces a general method of offset curve construction with tool-adaptive offsets. The offset path is obtained as a family of isolines of an anisotropic distance function relative to a seed curve on the workpiece surface. A distance function is defined on the surface and gives for every surface point its shortest distance to a given curve, the source curve. Anisotropy is defined by a metric tensor field on the surface. An isoline is defined by all those surface points whose distance is equal to a given real iso-value which defines the amount of offset against the seed curve.

The method is worked out for workpiece surfaces represented by triangular meshes. Its usefulness is demonstrated on the problem of varying cusp heights for milling and on the task of achieving a desired coating height for spray coating. The latter employs the path-geometry-last concept of Chapter 2 and presents an approach for its second step – the construction of a spray gun path.

1.4.3 Quantitative Improvement of Tool Impact Paths Defined by Isolines of Scalar Functions on Triangular Mesh Workpiece Surfaces

Isolines of distance functions suffer from two properties which have a negative influence on the usefulness of the resulting curves: locations with discontinuous derivatives and local extrema. These properties may induce sharp corners in isolines and varying distances, and a non-uniformly nested topological structure of isolines (cf. Fig. 3.2(a)), respectively. Chapter 4 is devoted to new optimization-based, quantitative approaches for contour-parallel and direction-parallel offset curves, respectively, to reduce these difficulties. For the contour-parallel case the curvature, the mutual distance, and the topology of the isolines are optimized over a finite-dimensional family of scalar functions derived from the distance function of the contour. In the direction-parallel case objectives including the number, the normal and the geodesic curvature of isolines are optimized over the distance functions of a finite-dimensional family of seed curves. Algorithms to solve these optimization problems on triangular meshes are proposed and employed to demonstrate the usefulness of the methods.

1.4.4 Algorithmic Aspects of Manipulator Placement Optimization for Path-Based Manufacturing Processes

The problem of manipulator and path placement is to find a placement of a manipulator so that it is able to execute a given tool path in an optimized way with respect to certain constraints and objectives. In Chapter 5, two approaches of optimization are employed to solve this problem. The first approach is to separate the placement and path optimization. For manipulators without redundant axes, a wide spectrum of existing optimization methods, including systematic sampling, evolution, and local descent is analyzed for suitability to the problem on test cases from spray coating. It turns out that evolutionary methods show the best performance with respect to the quality of the result and the computation time.

The second approach directly combines manipulator path optimization with the placement. Two novel methods are presented which replace either the static position of the workpiece or the strict path-following of the tool path, by an objective (soft constraint) in order to facilitate the optimization. The relaxed problems have the structure of a path optimization problem with a manipulator with redundant axes. A solution based on a quasi-Newton method is presented. The evaluation shows that the approach is practically useful.

1.5 Contributions by the Author

The novel approaches and methods and their analysis with respect to effectivity and efficiency are original work by the author of the thesis. Much of the content of chapters 2, 3, and 4 has been published as peer-reviewed journal papers, co-authored by the author of the thesis and the advisor of the thesis, Prof. Dr. Heinrich Müller [62, 63, 64]. The advisor directed the attention of the author to the problems tackled in the thesis, supported by discussions of specific technical aspects, and gave hints with respect to the presentation of the results.

1.6 Acknowledgement

The work on this thesis has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 708 “3D-Surface Engineering of Tools for Sheet Metal Forming – Manufacturing, Modeling, Machining”, project B4.

2 Tool Pose Optimization for Spray Deposition Processes

The aim of path planning for spray deposition processes is to find a time-dependent sequence of spray gun configurations so that a coating of desired height is achieved when executing the sequence. In this chapter, a novel approach to solve the planning task, called "path-geometry-last", is presented, which leads to a more general gun configuration cover problem. The gun configuration cover problem is to find a finite set of spray gun configurations, which minimize the error between a target coating and the coating induced by simultaneously activating those configurations. A suitable objective function for gun configuration covers is defined and an algorithmic solution for the optimization problem is presented. An experimental evaluation shows that good approximations of the desired coatings can be achieved within reasonable computation times. In contrast to other approaches, the path-geometry-last variant gains additional flexibility required to find complex paths for free-form workpieces.

2.1 Introduction

For spray coating, which is in the focus of this chapter, a spray gun is moved along a path above the workpiece. In each configuration, which is defined by the position and orientation of the spray gun relative to the workpiece, coating material is transferred from the spray gun onto the workpiece. The desired effect on the workpiece is to minimize the error between the achieved coating height and the target height.

In path planning, the geometry of the path is usually more or less defined depending on the geometry of the workpiece, the tool, and the manufacturing machine. Afterwards, the path is attributed with process-related parameters like material properties of the workpiece or the tool, which in particular have implications on physical aspects such as speed or acceleration. Finally, an adaptation of the path geometry is sometimes performed for reasons of optimization, or in order to get a feasible path at all. However, the potential of adaptation is restricted by the initial choice of the path geometry. This approach of "*path-geometry-first*" can be observed in milling where fixed path patterns like direction-parallel or contour-parallel pattern are widely used, but applies to spray coating as well [2, 35, 87]. Spray coating differs from contact-oriented manufacturing methods like milling in that the applicator, that is the spray gun, does not touch the surface. This causes considerably more flexibility for the geometric path than for example in milling where the path must tightly follow the workpiece surface. This fact makes the application of the geometry-first strategy to spray coating more difficult than for milling, at least if the additional flexibility is not heuristically reduced.

In this chapter, an alternative "*path-geometry-last*" strategy is proposed. It is based on the observation that a path can be represented by a set of configurations of the manufacturing unit or tool "covering" the workpiece. The idea of geometry-last path planning is to start with the calculation of an optimal cover of the workpiece with a finite set of configurations. Then the desired path is constructed using those configurations. The advantage of the path-geometry-last approach is that process-related parameters can be taken into account in the definition of the path. This is of particular relevance for spray coating with spray guns whose material flow rate is fixed and cannot be controlled during the process. In this case, using geometric degrees of freedom of the path is useful or even mandatory. Other authors also have taken the requirement of path adaptation explicitly into account (cf. Section 2.2), but either observed high computational requirements, or restricted the paths to very special path patterns like a sequence of parallel path segments which limit the cases to which it can be applied. In contrast, the geometry-last approach outlined in this chapter works for arbitrary paths on free-form surfaces.

A central component of the path-geometry-last strategy is the so-called *tool configuration cover problem*. In the case of spraying, the tool configuration cover problem is to find a finite set of spray guns located over the workpiece surface, which minimizes the error between a target coating and the coating induced by simultaneously activating these guns. The formulation and solution of the tool configuration cover problem for spray coating – for which it is particular useful – is the main contribution of this chapter.

The tool configuration cover problem for spray coating is specified as an optimization problem whose objective is the minimization of the error between the achieved and the desired coating height. The experimental analysis shows that low errors can be achieved at reasonable computation times.

Section 2.2 gives a survey of related work. Section 2.3 introduces the method of "path-geometry-last" and the gun configuration cover problem for spray coating, and gives a survey of the approach of solution to the gun configuration cover problem. The subsequent sections are devoted to its details. The results of an experimental analysis of the proposed approaches of optimization are presented in Section 2.8, and conclusions are given in Section 2.9.

2.2 Related Work

In contrast to path planning for e.g. milling, only little has been published on path planning for thermal spray coating in the scientific literature. Path planning for spraying is mentioned mainly in conjunction with spray painting [6, 7, 24, 28]. Two main approaches can be distinguished. The first one is automatic online path planning without a workpiece model, based on sensor information, e.g. optical and distance information [104]. The second one is offline path planning based on a CAD workpiece model [2, 6, 7, 28, 35, 59, 87].

For the second approach, which is in the focus of this chapter, besides intuitive heuristics, several solutions based on formulations as an optimization problem have been proposed. Kim and Sarma [59] give a comprehensive formulation as an optimization problem and discuss potential approaches to its solution. The optimization problem can be computationally complex, for example NP-hard, and its exact algorithmic solution can be extremely time-consuming and not possible in practice [2, 59]. For that reason, heuristic and approximate methods are of interest from the view of application. Besides local improvement as an optimization approach, Kim and Sarma [59] interpret path planning as a problem of calculus of variations [109] and outline methods to transfer it to a finite-dimensional optimization problem. Antonio [2] presents two versions of problem specifications, one assuming a fixed path geometry and one assuming a class of feasible trajectories specified by constraints. Both versions are converted into nonlinear programming problems, and it is noted for the second case that the computational requirements are too high in practice. Ramabhadran and Antonio [87] fix the path geometry and give formulations of objective functions which lead to either linear or quadratic programming problems which can be solved up to three orders of magnitude faster than the general nonlinear program by Antonio et al. Duncan et al. [35] restrict the path geometry to the very specific pattern of parallel sweeping paths. This allows a very efficient solution, but is of limited use on complex free-form surfaces because of the special path pattern. Although being of the geometry-first type, a rule for the required density of the path is derived based on the coating profile. Atkar et al. [7] incrementally construct a sequence of curve segments by selecting a seed segment and iteratively offsetting it sideways within the surface to generate further segments which finally cover the surface. The distance between neighboring segments is chosen depending on the paint profile. This is an efficient approach which seems more flexible than the one by Duncan et al. [35], but it is still restricted to a special path pattern which, for more complex free-form surfaces, requires a segmentation of the surface into patches which are suited to be covered by this path pattern [6]. Besides parallel paths which are currently preferred for spray coating, alternative path patterns like the billiard paths by Jones et al. [55] and the contour-to-contour paths by Hegels and Müller [46] have been proposed for thermal spray coating where bounding the heat in the workpiece is an additional objective.

In contrast to those solutions, this chapter advocates the path-geometry-last type. It is more flexible than the path-geometry-first approaches, can be applied to arbitrary free-form surfaces, and its computational efficiency is high enough for application in practice. The following sections are devoted to one of its major parts, the gun configuration cover problem. The presentation is related to a publication of the author [62]. The main difference is the approach to solve the related optimization problem.

2.3 The Method

The spray coating problem can be formulated as follows:

Spray Coating Problem

Given: A workpiece surface, a desired coating height at every location of the surface, and a spray gun delivering coating material.

Wanted: A gun movement over the surface so that the resulting material deposition achieves the desired coating height.

The gun movement may be represented by a finite sequence of gun poses augmented with spray times, i.e. durations, which are called *gun configurations*. The spray times specify the speed of the movement. A continuous movement may be obtained by interpolation or approximation of the configurations.

A simple but important observation is that the effect of material deposition does not depend on the time arrangement of the gun configurations, at least if detailed physical effects are neglected. Hence, it is sufficient with respect to material deposition to consider a set of spray gun configurations, i.e. gun poses together with spray times. A set of this kind is called a *gun configuration cover*. This leads to the

Path-geometry-last Approach

1. Determine a gun configuration cover which achieves the desired coating height.
2. Construct a spray gun path from the gun configuration cover.

In the following, a solution of step 1, the gun configuration cover problem, is presented:

Algorithm 1 Solution of the gun configuration cover problem.

```
Input: A workpiece surface, a desired coating height at every location of the surface and a
    spray gun delivering coating material.
Output: A finite set of spray gun configurations which generate a coating height with
    minimal deviation from the desired coating height.
1: Configure the deposition model.
2: Set up the coating height representation and the gun configurations of the cover.
3: Set up the objective function.
4: Solve the optimization problem.
```

Gun Configuration Cover Problem

Given:	A workpiece surface, a desired coating height at every location of the surface, and a spray gun delivering coating material.
Wanted:	A finite set of spray gun configurations which generate a coating height with minimal deviation from the desired coating height.

The gun configuration cover problem avoids the optimization over sequential arrangements of spray guns which is inherent to the optimization over paths. Hence, a solution with significantly less computational requirements can be expected.

A further benefit is that tool configuration covers can also be applied to solve the second phase of geometry-first-based planning. This is performed by defining a candidate gun configuration cover as a finite set of time sampled points on the path from the first phase. The solution of the resulting gun cover problem yields modified gun positions which define a distortion of the path that takes into account the objective of achieving a desired coating height.

A drawback of the approach could be that the isolated tool configurations should be distributed uniformly over the surface in order to be equally suited for all potential paths. Hence, such a cover may influence the workpiece surface less than the continuous tool configurations of a path. However, this could be taken into account after path construction by applying the method once more for covering configurations derived from a finite set of time sampled points on the path, as just described.

The solution of the problem of gun configuration cover presented in the following consists of several steps which are displayed in Algorithm 1. First of all, a deposition model for spray coating is required in the first step. Such a model will be presented in Section 2.4. Step 2 is concerned with the representation of a gun cover and the coating height; the related details will be provided in Section 2.5. The gun cover representation will be geared towards an impact path-based approach of path construction. Chapter 3 will later-on work out such an approach for the special case of impact paths defined by isolines. Next, the objective function with respect to the desired coating height is set up in step 3. It will be formally presented in Section 2.6. Finally, the resulting optimization problem is solved in step 4, with the method described in Section 2.7.

2.4 Deposition Model

During a thermal spray coating process, molten particles are accelerated in a spray gun towards a prepared surface where they impact with a high velocity, flatten, and form a thin layer on the substrate [39]. A deposition model describes the effect of this process on the surface depending on parameters of the spray gun. The deposition model employed in step 1

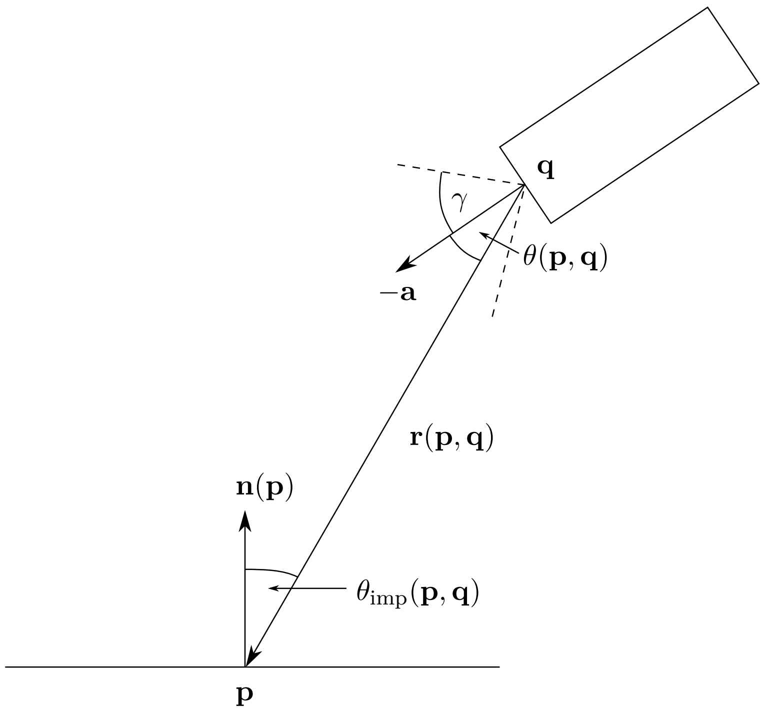

$\mathbf{q}$
γ
$\theta(\mathbf{p},\mathbf{q})$
$-\mathbf{a}$
$\mathbf{r}(\mathbf{p},\mathbf{q})$
$\mathbf{n}(\mathbf{p})$
$\theta_{\text{imp}}(\mathbf{p},\mathbf{q})$
$\mathbf{p}$

Fig. 2.1: The geometric parameters of the deposition model of Duncan et al. [35].

of the algorithm is based on the model of Duncan et al. [35] which is a typical representative of such models. In this model, the particles fly on straight rays sharing a starting point $\mathbf{q}$ inside the gun, called *gun reference point* in the following. The resulting spray cone is assumed to be symmetric so that it is sufficient to specify the flow rate along a ray with respect to the angle $\theta(\mathbf{p},\mathbf{q})$ between the direction $\mathbf{r}(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|$ of the ray to the sample point $\mathbf{p}$ and the unit direction $-\mathbf{a}$ of the center line of the cone, called *spray angle*. The spray angle is limited by the *opening angle* γ of the cone. Moreover, Duncan et al. model the reflection of material on the surface depending on the angle $\theta_{\text{imp}}(\mathbf{p},\mathbf{q})$ between the inverse ray vector $-\mathbf{r}(\mathbf{p},\mathbf{q})$ and the surface normal $\mathbf{n}(\mathbf{p})$ at $\mathbf{p}$, called *impact angle*. Reflection is of particular importance for metal spray coating. Figure 2.1 illustrates those parameters.

The behavior of the spray gun is described by the *flow rate distribution function* $f(\mathbf{p},\mathbf{q})$ with unit [mm^{-2}]. According to Duncan et al. [35],

$$f(\mathbf{p},\mathbf{q}) = \frac{\cos(\theta_{\text{imp}}(\mathbf{p},\mathbf{q})) \cdot \Theta(\theta(\mathbf{p},\mathbf{q}),\gamma) \cdot \zeta(\theta_{\text{imp}}(\mathbf{p},\mathbf{q}))}{\|\mathbf{r}(\mathbf{p},\mathbf{q})\|^2}$$

is used, where the function Θ represents the droplet distribution for the spray angle θ per solid angle, and ζ provides the fraction of material that is not reflected from the surface. The coating height, dh, generated within a spray time dt is depending on the volume flow rate V_{T} [mm^3/s] of material of the gun and the visibility of the surface point $\mathbf{p}$ from the gun reference point $\mathbf{q}$,

$$\mathrm{d}h(\mathbf{p}) = f(\mathbf{p},\mathbf{q}) \cdot \delta(\mathbf{p},\mathbf{q}) \cdot V_{\mathrm{T}} \cdot \mathrm{d}t\,, \tag{2.1}$$

where $\delta(\mathbf{p},\mathbf{q})$ denotes the visibility term which is equal to 1 if mutual visibility is given, and equal to 0 otherwise.

This representation differs from the one by Duncan at al. in that the volume per area, i.e. the height, instead of mass per area is considered. The model of Duncan et al. results by replacing $\mathrm{d}h$ by a mass per area $\mathrm{d}m$ and V_T by a mass flow rate u. This is achieved by multiplying V_T and $\mathrm{d}h$ with the mass density ρ_mass of the coating material. The latter can be understood as $\mathrm{d}h \cdot \rho_\mathrm{mass} = \mathrm{d}V/\mathrm{d}M \cdot \rho_\mathrm{mass} = \mathrm{d}m$.

2.5 Gun Configuration Covers and Deposit Representation

The aim of this section is to introduce the concept of gun configuration covers formally. It starts with a continuous version (Section 2.5.1) where the deposition height at a surface point generated by a gun cover is specified by an integral over the guns. By approximating the integrand in a piecewise constant manner, a discrete version can be derived (Section 2.5.2). The discretization leads to the definition of a concept of discrete gun covers on surfaces represented by triangular meshes (Section 2.5.3). This representation is set up in step 2 of the algorithm. A further aspect is the definition of the spray directions of the guns (Section 2.5.4), which is also part of the preparation of a gun cover in step 2.

The introduction of the concept of continuous gun covers helps to treat re-discretizations of discrete gun covers as required in Section 3.6.1 by interpreting them as different discretizations of a common continuous gun cover.

2.5.1 Continuous Gun Configuration Covers

A gun cover consists of a family of guns. Every gun has a configuration which consists of its location, its volume flow rate, and its spray time density.

The geometric gun configuration $\mathbf{g}(\mathbf{m})$ in a gun cover is parameterized over the workpiece surface M which is assumed to be a manifold surface M. The geometric configuration is specified in terms of the distance $d(\mathbf{m})$ of the gun to a gun base point $\mathbf{m} \in M$ and a gun direction given by the location of the center line of the spray cone as a unit vector, i.e. the spray direction $\mathbf{a}(\mathbf{m})$ (Fig. 2.1):

$$\mathbf{g}(\mathbf{m}) = (\mathbf{a}(\mathbf{m}), d(\mathbf{m}))\,. \tag{2.2}$$

The point

$$\mathbf{q}(\mathbf{m}) = \mathbf{m} + \mathbf{a}(\mathbf{m}) \cdot d(\mathbf{m}) \tag{2.3}$$

is the reference point which lies at the tip of the spray cone. The gun reference point together with the vector $-\mathbf{a}(\mathbf{m})$ define a *partial gun reference frame*. The spray cones are assumed to be symmetric so that it can be completed arbitrarily to complete tool frame (cf. Section 5.3.2).

The reason for this gun representation is the subsequent impact path-based path construction. An impact path is a path on the workpiece surface which represents the impact of a production tool on the surface along its tool path. Figure 3.1(b) sketches the relation between a gun path and its impact path for spray coating. The details of path construction from a gun cover will be presented in Chapter 3. Furthermore, if a spray path candidate is described relative to a given impact path, this approach also allows its optimization in the sense of the geometry-first case.

A drawback of the surface-bound parametrization is that surface points near the surface boundary are not covered by as many guns as points far from the boundary. Thus, the desired coating height might not be achievable. An approach to cope with this problem used here is to virtually extend the surface at the boundary so that the guns can also be positioned beyond the original surface.

Let $V_\mathrm{T}(\mathbf{m},t)$ [mm^3/s] be the volume flow rate and $t_M(\mathbf{m})$ [s/mm^2] be the spray time density of the gun $\mathbf{g}(\mathbf{m})$. $V_\mathrm{T}(\mathbf{m},t)$ is assumed to have a constant value independent from location $\mathbf{m}$ and time t, i.e. $V_\mathrm{T}(\mathbf{m},t) = V_\mathrm{T}$. With the deposition model (2.1), the contribution of gun $\mathbf{g}(\mathbf{m})$ to the coating height at a surface point $\mathbf{p}$ is

$$h_M(\mathbf{p},\mathbf{m}) = \int_0^{t_M(\mathbf{m})} f(\mathbf{p},\mathbf{q}(\mathbf{m})) \cdot \delta(\mathbf{p},\mathbf{q}(\mathbf{m})) \cdot V_\mathrm{T}\,\mathrm{d}t\,. \tag{2.4}$$

h_M [mm/mm^2] denotes the coating height density. The coating height density and the spray time density are related to a differential surface element $\mathrm{d}M(\mathbf{m})$. Equation (2.4) can be rewritten as

$$h_M(\mathbf{p},\mathbf{m}) = f(\mathbf{p},\mathbf{q}(\mathbf{m})) \cdot \delta(\mathbf{p},\mathbf{q}(\mathbf{m})) \cdot V_M(\mathbf{m})\,, \tag{2.5}$$

where

$$V_M(\mathbf{m}) = \int_0^{t_M(\mathbf{m})} V_\mathrm{T}\,\mathrm{d}t = V_\mathrm{T} \cdot t_M(\mathbf{m})$$

is the material flow density with unit [mm^3/mm^2]. This leads to the following alternative version of Eq. (2.5):

$$h_M(\mathbf{p},\mathbf{m}) = f(\mathbf{p},\mathbf{q}(\mathbf{m})) \cdot \delta(\mathbf{p},\mathbf{q}(\mathbf{m})) \cdot V_\mathrm{T} \cdot t_M(\mathbf{m})\,. \tag{2.6}$$

The coating height (unit [mm]) generated by all guns of the gun cover at a point $\mathbf{p}$ results from integration over all guns:

$$h(\mathbf{p}) = \int_M f(\mathbf{p},\mathbf{q}(\mathbf{m})) \cdot \delta(\mathbf{p},\mathbf{q}(\mathbf{m})) \cdot V_T \cdot t_M(\mathbf{m})\,\mathrm{d}M(\mathbf{m})\,. \tag{2.7}$$

2.5.2 Discretization

The continuous case is made discrete by decomposing the surface M into a finite number of surface elements M_j, and choosing points $\mathbf{m}_j \in M_j$, $j = 0,\ldots,n$. The points define a *finite set $\mathcal{G}$ of gun configurations*. The elements should be selected uniformly in shape, size and location in order to achieve a good approximation.

The coating height is represented at a *finite set $\mathcal{P}$ of height sample points* $\mathbf{p}_i$, $i = 0,\ldots,m$, on the surface. The approximation of the integral (2.7) in a piecewise constant manner yields the coating height of a discrete gun configuration cover,

$$\begin{aligned} h(\mathbf{p}_i) &= \sum_{j=0}^{n} \int_{M_j} f(\mathbf{p}_i,\mathbf{q}(\mathbf{m})) \cdot \delta(\mathbf{p}_i,\mathbf{q}(\mathbf{m})) \cdot V_\mathrm{T} \cdot t_M(\mathbf{m})\,\mathrm{d}M(\mathbf{m}) \\ &\approx \sum_{j=0}^{n} f_{i,j} \cdot \delta_{i,j} \cdot V_\mathrm{T} \cdot t_{M,j} \cdot A_j\,, \end{aligned}$$

where A_j is the area of M_j, $f_{i,j} := f(\mathbf{p}_i,\mathbf{q}(\mathbf{m}_j))$, $\delta_{i,j} := \delta(\mathbf{p}_i,\mathbf{q}(\mathbf{m}_j))$, and $t_{M,j} := t_M(\mathbf{m}_j)$. Defining

$$\rho_j = 1/A_j \tag{2.8}$$

as the *gun sampling density* at $\mathbf{m}_j$ and

$$t_j = t_{M,j} \cdot A_j = \frac{t_{M,j}}{\rho_j} \tag{2.9}$$

as the *spray time* results in

$$h(\mathbf{p}_i) = \sum_{j=0}^{n} f_{i,j} \cdot \delta_{i,j} \cdot V_{\mathrm{T}} \cdot \frac{t_{M,j}}{\rho_j} \tag{2.10}$$

$$= \sum_{j=0}^{n} f_{i,j} \cdot \delta_{i,j} \cdot V_{\mathrm{T}} \cdot t_j \, . \tag{2.11}$$

2.5.3 Discrete Gun Configuration Covers on Triangular Meshes

In this thesis, triangular meshes are used as representation of the workpiece surface for discrete gun configuration covers. One reason is that they are well suited for calculation purposes, for example for the evaluation of the visibility term of the deposition model (2.1).

A *triangular mesh* is composed of triangles, edges, and vertices. Two intersecting triangles share a vertex or an edge. Each edge has at most two and at least one incident triangles. An edge with exactly one incident triangle is on the boundary of the surface defined by the mesh. Figure 3.3 shows two surfaces represented by triangular meshes. Workpieces represented by meshes can be obtained by conversion of other computer-based geometry representations, like e.g. NURBS, or from scan data [67], but also by direct mesh-based design [17].

The resolution of the mesh has to be chosen depending on the application or on algorithmic requirements. Methods exists for adaptive refinement and coarsening of given meshes [17]. In the given case, the vertex set of the mesh is used for the set $\mathcal{G}$ of gun locations and the set $\mathcal{P}$ of height sampling points. In order to transform arbitrarily given workpiece meshes into well-suited meshes, an adaptation of the remeshing algorithm of Turk [102] is proposed in the following. It starts by distributing points on the given mesh by randomly choosing a face of the mesh and then randomly setting a point on that face. The probability for each face to be chosen is proportional to its area. Afterwards a repelling force is calculated between all neighboring points so that the points are pushed away from each other. In this way a uniform distance between the points is achieved.

In contrast to Turk, the point movement is constrained to the mesh and implemented as a straightest geodesic, instead of projecting the points back to the mesh after each step. A straighest geodesic is some sort of locally shortest path on a mesh [86]. The movement direction of each point is therefore projected onto the tangential plane at first, and then used as starting direction for the straightest geodesic. Problems arise at the boundary of a mesh, which are solved by letting the points slip along the boundary edges.

The most time-consuming part of the algorithm is to find the neighbors of every point. This task is efficiently supported in a common way by hashing on a regular spatial grid in order to reduce the number of distance tests. Spatial hashing assigns a point to the grid cell in which it lies.

By basically selecting both the set $\mathcal{G}$ of gun locations and the set $\mathcal{P}$ of height sampling points as the set of vertices of the workpiece surface mesh, all sampling points are covered by guns. However, a different choice is possible, for example by restricting $\mathcal{G}$ to a subset of the vertex set or by using two different meshes altogether. The height sample points can even be arbitrary points on the surface mesh instead of just vertices.

The surface elements M_i are chosen as the Voronoi regions around the vertices $\mathbf{v}_i$ [79], cf. Fig. 2.2. With angles $\alpha_{i,j}$, $\beta_{i,j}$ and $N_1(i)$ the 1-ring neighborhood of vertex $\mathbf{v}_i$ as defined in the figure, the area of M_i is

$$A_i = \frac{1}{8} \sum_{j \in N_1(i)} (\cot \alpha_{i,j} + \cot \beta_{i,j}) \|\mathbf{v}_i - \mathbf{v}_j\|^2 \, . \tag{2.12}$$

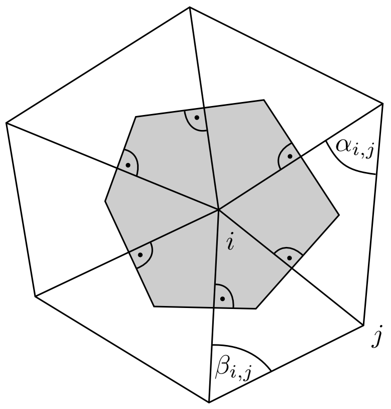


Fig. 2.2: 1-ring neighborhood of vertex i with its Voronoi area and angle $\alpha_{i,j}$ and $\beta_{i,j}$ corresponding to the edge between vertex $\mathbf{v}_i$ and $\mathbf{v}_j$.

This holds for meshes with non-obtuse angles. The Voronoi regions satisfy the requirements of uniform shape.

2.5.4 Definition of Spray Directions

In general, spray directions $\mathbf{a}(\mathbf{m})$ orthogonal to the surface at $\mathbf{m}$ are preferred. Thus, the surface normals at the gun base points are a good choice for spray directions. For triangular meshes, normals at the vertices, which represent the normals of the smooth surface, are required. Such normals may be provided when converting a smooth surface to a mesh. Otherwise, vertex normals may be estimated heuristically directly from the mesh [17]. For instance, some sort of weighted mean of the normals of the triangles incident to a vertex could be taken.

The spray directions defined in this manner may change considerably at locations of high curvature. Such rapid changes might not be favorable for spray paths derived from the gun configuration cover. Another aspect is that the impact angles in the area under the spray cone should be as small as possible, so that a local orthogonal spray direction is not sufficient. These drawbacks can be diminished by smoothing the normals. An approach which allows to control the degree of smoothness is smoothing by diffusion [32]. Smoothing by diffusion is based on the paradigm of physical diffusion processes. Let $\mathbf{f}$ be a time- and space-dependent function in a spatial domain which defines the property being subject to diffusion. The diffusion process is described by the partial differential equation

$$\frac{\partial \mathbf{f}}{\partial t} = c \cdot \Delta \mathbf{f}\,,$$

and an initial property distribution $\mathbf{f}(.,t_0)$ at the start time t_0 of diffusion. $c > 0$ describes the strength of diffusion. Δ denotes the Laplace operator. In physics, for example, $\mathbf{f}(\mathbf{p},t)$ could be the temperature at a location $\mathbf{p}$ at a time t, and c could specify the degree of heat diffusion. On triangular meshes, the function $\mathbf{f}$ is given at the mesh vertices, and the discrete Laplace-Beltrami operator in the cotangent version [79] given by

$$\Delta_M \mathbf{f}_i = \frac{1}{2A_i} \sum_{j \epsilon N_1(i)} (\cot \alpha_{i,j} + \cot \beta_{i,j})(\mathbf{f}_j - \mathbf{f}_i)$$

is applied (cf. Fig. 2.2). The equation of diffusion is numerically solved by transferring it into a discrete version which leads to the iterative scheme

$$\mathbf{f}_i^{n+1} = (1 + \Delta t \cdot c \cdot \Delta_M) \cdot \mathbf{f}_i^n\,,\ i = 1, \ldots, m,$$

where Δt is the time step, m the number of vertices, and $\mathbf{f}_i^0$ the given start function. The size of the time step Δt, the size of the diffusion coefficient c, and the number of iterations influence the strength of smoothing. Explicit integration of this sort has the disadvantage that the time step is bound to keep the method stable, which may result in a longer runtime. A more favorable implicit formulation of the iteration step is [32]

$$(1 - \Delta t \cdot c \cdot \Delta_M) \cdot \mathbf{f}_i^{n+1} = \mathbf{f}_i^n\,,\ i = 1, \ldots, m.$$

Desbrun et al. [32] suggest a normalized version of this equation, in which the cotangents weights sum up to one and the coefficient $1/(2A_i)$ is omitted. In this way, the explicit integration scheme is more stable and a step length from the interval $[0, 1]$ can be used.

In the application of normal smoothing the $\mathbf{f}_i^0$ denote the given normals at vertices i.

2.6 Objective Function

The *aim of optimization* of a gun configuration cover is to generate a coating on a given surface with a target height distribution. The desired height values are provided at the set of sampling points. The optimization variables are the distance to the surface and the spray time density of every gun of the cover. This makes the position of the gun a variable to be optimized, in contrast to the geometry-first approaches [35, 87], where the positions of the guns are fixed on a given path. Optimizing the position is of special interest for processes where the flow rate of the gun is fixed, as considered here.

Those requirements are formally expressed by the objective function

$$\begin{aligned} Z(d_0, \ldots, d_n, t_{M,0}, \ldots, t_{M,n}) = {} & C_e \cdot Z_\mathrm{e}(d_0, \ldots, , d_N, t_{M,0}, \ldots, t_{M,n}) \\ & + C_\mathrm{d} \cdot Z_\mathrm{d}(d_0, \ldots, d_n) \\ & + C_\mathrm{t} \cdot Z_\mathrm{t}(t_{M,0}, \ldots, t_{M,n}) \\ & + C_\mathrm{sd} \cdot Z_\mathrm{sd}(d_0, \ldots, d_n)\,. \end{aligned} \tag{2.13}$$

which is set up in step 3 of the algorithm. d_j and $t_{M,j}$ denote the distance and the spray time density of gun j, $j = 0, \ldots, n$, respectively.

The term Z_e describes the height error over all height sample points i, $i = 0, \ldots, m$, in terms of the height of the sprayed layer, cf. Eq. (2.10) and of the target height, h_i^s. The error is expressed as the mean of the squares of the differences, normalized by the mean h_mean of the h_i^s,

$$Z_\mathrm{e}(d_0, \ldots, d_n, t_{M,0}, \ldots, t_{M,n}) = \frac{1}{m} \sum_{i=0}^{m} Z_{\mathrm{e},i}(d_0, \ldots, d_n, t_{M,0}, \ldots, t_{M,n})^2, \tag{2.14}$$

$$Z_{\mathrm{e},i}(d_0, \ldots, d_n, t_{M,0}, \ldots, t_{M,n}) = \frac{h_i^s - \sum_{j=0}^{n} (f_{i,j}(d_j) \cdot \delta_{i,j} \cdot V_\mathrm{T} \cdot t_{M,j}/\rho_j)}{h_\mathrm{mean}},\ i = 0, \ldots, m\,. \tag{2.15}$$

The variables d_j and $t_{M,j}$ are subject to the constraints

$$d_\mathrm{min} \le d_j \le d_\mathrm{max}\,, \tag{2.16}$$

$$t_{M,j} > 0\,, \tag{2.17}$$

where $d_{\min}$ and $d_{\max}$ are the minimum and maximum distance. These constraints can be easily integrated into the optimization method.

However, solely optimizing the height might lead to solutions that are not applicable. A constraint which is typical for many spray systems is that the local variation of the distances and spray times should be smooth. The reason is that extreme changes of the gun position or velocity along a path potentially cannot be executed. For that reason smoothness terms for the distance, Z_d, and spray time density, Z_t, are added to the objective function weighted by C_d and C_t,

$$Z_{\mathrm{d}}(d_0,\ldots,d_n) = \frac{1}{n}\sum_{j=0}^{n} \rho_j \cdot Z_{\mathrm{d},j}(d_0,\ldots,d_n)^2, \tag{2.18}$$

$$Z_{\mathrm{d},j}(d_0,\ldots,d_n) = \frac{d_j - d_{j,\mathrm{int}}}{d_{\mathrm{s}}},\ j = 0,\ldots,n, \tag{2.19}$$

$$Z_{\mathrm{t}}(t_{M,0},\ldots,t_{M,n}) = \frac{1}{n}\sum_{j=0}^{n} \rho_j \cdot Z_{\mathrm{t},j}(t_{M,0},..,t_{M,n})^2, \tag{2.20}$$

$$Z_{\mathrm{t},j}(t_{M,0},\ldots,t_{M,n}) = \frac{1/t_{M,j} - v_{j,\mathrm{int}}}{v_s},\ j = 0,\ldots,n. \tag{2.21}$$

The first term consists of the squares of the differences of d_j to an interpolated distance value $d_{j,\mathrm{int}}$ at gun j, normalized by a given constant desired distance d_{s}. The idea is that the interpolated values, i.e. $d_{j,\mathrm{int}}$, can be considered as the result of a local low-pass filtering. Hence, the difference $(d_j - d_{j,\mathrm{int}})^2$ can be seen as high-pass filter value whose amount is lower for a smoother function.

The second term consists of the squares of the differences of $1/t_{M,j}$ to a value $v_{j,\mathrm{int}}$ interpolating the $1/t_{M,k}$ values at gun j, normalized by a constant value v_s. In the optimization process of Section 2.7, v_s is taken as the mean of all $1/t_{M,j}$ of the initialization step.

The interpolations at gun j are performed on the 1-ring neighborhood $N_1(j)$ of vertex j using the cotangent weights from Fig. 2.2,

$$w_j = \frac{1}{\sum_{k\in N_1(j)}(\cot\alpha_{j,k} + \cot\beta_{j,k})} \sum_{k\in N_1(j)} (\cot\alpha_{j,k} + \cot\beta_{j,k}) \cdot w_k\,,$$

where the w_k denote the values to be interpolated. In contrast to uniform weights the cotangent weights take the size and shape of the surrounding triangles into account.

Finally, the term

$$Z_{\mathrm{sd}}(d_0,\ldots,d_N) = \frac{1}{n}\sum_{j=0}^{n} Z_{\mathrm{sd}}(d_j)^2,\ Z_{\mathrm{sd}}(d_j) = \frac{d_j - d_s}{d_s}\,, \tag{2.22}$$

representing the deviation from the given desired distance d_s, is added to the objective function for regularization purposes.

2.7 Optimization

Various iterative numerical optimization methods can be applied to non-linear least-squares problems [84], e.g. the conjugate gradient method, the BFGS quasi-Newton method, or methods like Newton, Gauss-Newton or Levenberg-Marquardt. The most popular one is the Levenberg-Marquardt algorithm (LM algorithm) which will be employed here. In the paper [62] by the author, an alternative gradient-descent approach has been presented. In the experimental analysis it has turned out as computationally faster, but the quality of the results has been minor.

Specifically for this problem, it is essential to make the LM method independent of the scale of the variables. The LM algorithm provides a method to achieve this, by using the diagonal of the matrix $J^\top J$, J the Jacobi matrix of the objective function, instead of the identity matrix for regularization [84].

The application of the LM algorithm requires a slight reformulation of the objective function (2.13):

$$Z(d_0,\dots,d_n,t_{M,0},\dots,t_{M,n}) = \left\|\hat{\mathbf{Z}}(d_0,\dots,d_n,t_{M,0},\dots,t_{M,n})\right\|_2^2$$

with

$$\begin{aligned}\hat{\mathbf{Z}}(d_0,\dots,d_n,t_{M,0},\dots,t_{M,n})^\top = \Big(&\hat{Z}_{\mathrm{e},j}(d_0,\dots,d_n,t_{M,0},\dots,t_{M,n})|_{i=0,\dots,m}\,,\\ &\hat{Z}_{\mathrm{d},j}(d_0,\dots,d_n)|_{j=0,\dots,n}\,,\\ &\hat{Z}_{\mathrm{t},j}(t_{M,0},\dots,t_{M,n})|_{j=0,\dots,n}\,,\\ &\hat{Z}_{\mathrm{sd}}(d_j)|_{j=0,\dots,n}\Big)\end{aligned}$$

and

$$\begin{aligned}\hat{Z}_{\mathrm{e},i} &= \sqrt{\frac{C_e}{m}}\cdot Z_{\mathrm{e},i}\,,\\ \hat{Z}_{\mathrm{d},j} &= \sqrt{\frac{C_d\rho_j}{n}}\cdot Z_{\mathrm{d},j}\,,\\ \hat{Z}_{\mathrm{t},j} &= \sqrt{\frac{C_t\rho_j}{n}}\cdot Z_{\mathrm{t},j}\,,\\ \hat{Z}_{\mathrm{sd},j} &= \sqrt{\frac{C_{sd}}{n}}\cdot Z_{\mathrm{sd},j}\,.\end{aligned}$$

The partial derivatives of the components of $\hat{\mathbf{Z}}$ needed for the rows of the Jacobian matrix in every iteration step of the LM algorithm are

$$\begin{aligned}\frac{\partial\hat{Z}_{\mathrm{e},i}}{\partial d_k} &= \sqrt{\frac{C_e}{m}}\cdot\left(-\frac{1}{h_{\mathrm{mean}}}\cdot\frac{\partial f_{i,k}(d_k)}{\partial d_k}\cdot V_{\mathrm{F}}\cdot t_{M,k}/\rho_k\right),\\ \frac{\partial\hat{Z}_{\mathrm{d},j}}{\partial d_k} &= \sqrt{\frac{C_d\rho_j}{n}}\cdot\frac{1}{d_s}\cdot\left(\frac{\partial d_j}{\partial d_k}-\frac{\partial d_{j,\mathrm{int}}}{\partial d_k}\right),\\ \frac{\partial\hat{Z}_{\mathrm{sd},j}}{\partial d_k} &= \sqrt{\frac{C_{sd}}{n}}\cdot\frac{1}{d_s}\cdot\frac{\partial d_j}{\partial d_k}\,,\\ \frac{\partial\hat{Z}_{\mathrm{e},i}}{\partial t_{M,k}} &= \sqrt{\frac{C_e}{m}}\cdot\left(-\frac{1}{h_{\mathrm{mean}}}\cdot f_{i,k}(d_k)\cdot V_{\mathrm{F}}/\rho_k\right),\\ \frac{\partial\hat{Z}_{\mathrm{t},j}}{\partial t_{M,k}} &= \sqrt{\frac{C_t\rho_j}{n}}\cdot\frac{1}{v_s}\cdot\left(-\frac{1}{t_{M,k}^2}-\frac{\partial v_{j,\mathrm{int}}}{\partial t_{M,k}}\right),\end{aligned}$$

Table 2.1: Coating errors for several basic surface shapes with a Gaussian distribution as desired coating height.

shape	root mean square error [%]	maximum coating error [%]
plane	0.017	0.068
saddle	0.021	0.066
cylinder	0.017	0.089
sphere	0.012	0.049
interior cylinder	0.015	0.108
interior sphere	0.017	0.115

and their computation is not more time-consuming than a normal function evaluation.

In the implementation, the derivatives of the flow rate distribution function $f_{i,k}(d_k)$ with respect to the distance are approximated by a difference quotient of two function values,

$$\frac{\partial f_{i,k}(d_k)}{\partial d_k} \approx \frac{f_{i,k}(d_k + \Delta d_k) - f_{i,k}(d_k)}{\Delta d_k} \, .$$

The evaluation of the objective function and its partial derivatives is restricted to those terms which are not excluded by the potentially limited spray cone extension and the mutual invisibility of the spray gun reference point $\mathbf{q}_j$ and the height sampling point $\mathbf{p}_i$, expressed by the factor $\delta_{i,j}$ in Eq. (2.15). The mutual visibility is calculated by ray casting. A line segment from $\mathbf{q}_j$ and to $\mathbf{p}_i$ is tested for collision with the surface. An efficient implementation is achieved by using a hierarchy of bounding volumes to subdivide the mesh. The calculation time grows linearly with the number of height sample points, which could make ray casting unsuitable for an extremely large number of sample points. However, in the use cases of the thesis (cf. Section 2.8), the ray casting approach has shown to be sufficient for the number of sample points used. Another approach to visibility tests is to use the graphics hardware which includes a visibility test based on depth buffers in the graphics pipeline. This approach has been employed by Kout et al. [62] in a previous solution of the gun cover problem.

Basically, the visibility test has to be applied in every iteration step of the LM algorithm. However, for guns whose visibility does not change when varying the distance, cf. Eq. (2.16), a visibility calculation in the initialization phase is sufficient. Many surfaces, in particular those with no, or just minor, concavities have this property everywhere.

The iterative solution process of the LM algorithm requires an initialization of d_j and $t_{M,j}$, $j = 0, \ldots, n$. The gun distances are set to the desired distance, i.e. $d_j = d_s$, $j = 0, \ldots, n$. The spray time densities are also set to a uniform value, i.e. $t_{M,j} = 1/v_s$, cf. Eq. (2.20). v_s is determined by an iteration process with fixed distances $d_j = d_s$ and the same variable t_M for the spray time density of every gun. The optimization problem is reduced to a single equation in one variable, which can be solved directly.

2.8 Analysis

The proposed optimization method has been computationally analyzed on a set of basic surface types typical for the local shapes of smooth free-form surfaces: “plane”, “cylinder”, “sphere”, and “saddle”. The models have approximately a size of 200×200 mm. Furthermore, the CAD model of a real workpiece has been used. It has a size of $340 \times 140 \times 60$ mm. The quality of approximation of the desired coating, the computation time, and insights into the

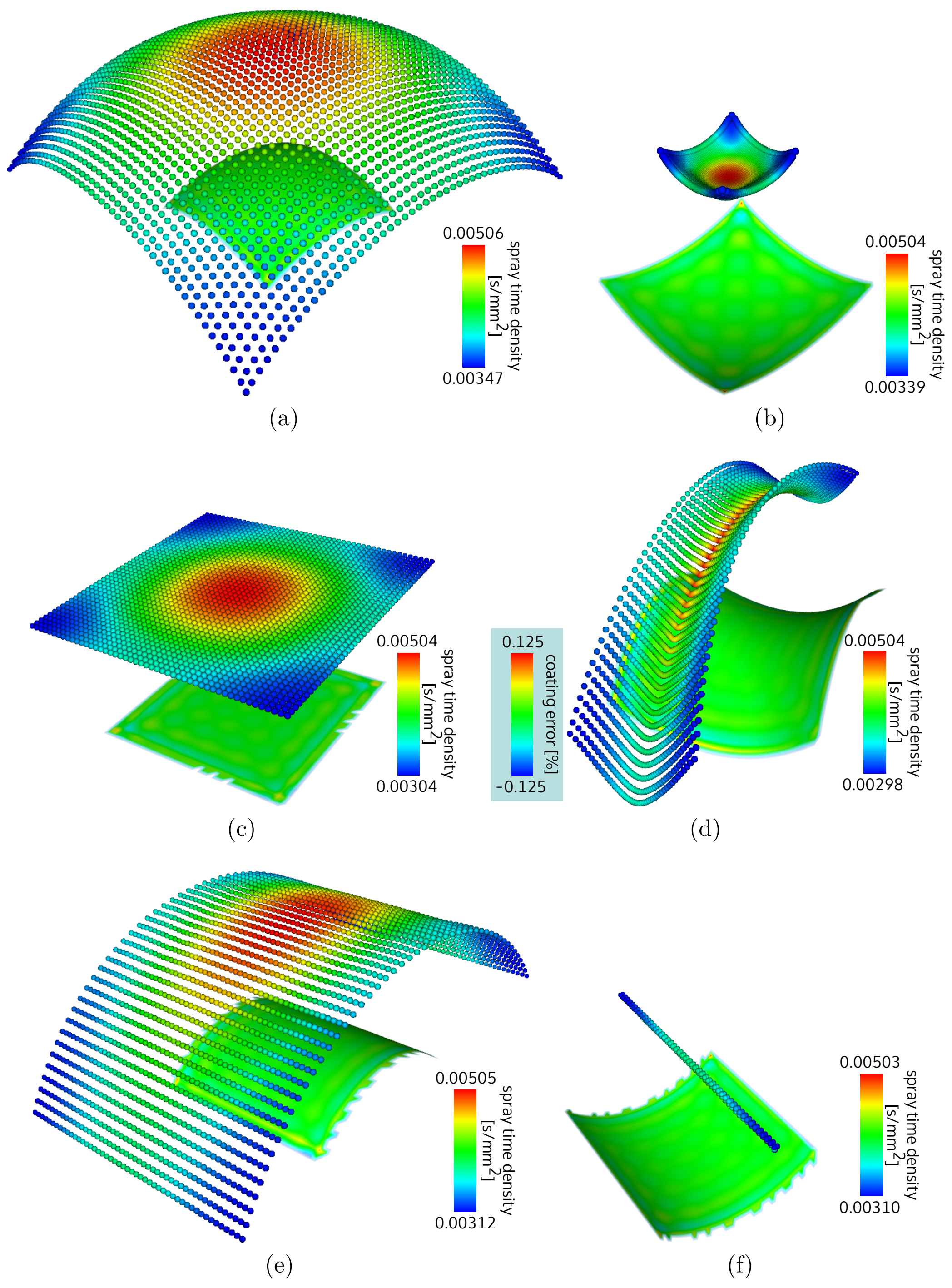

0.00506
spray time density [s/mm^2]
0.00347
(a)
0.00504
spray time density [s/mm^2]
0.00339
(b)
0.00504
spray time density [s/mm^2]
0.00304
(c)
0.125
coating error [%]
-0.125
0.00504
spray time density [s/mm^2]
0.00298
(d)
0.00505
spray time density [s/mm^2]
0.00312
(e)
0.00503
spray time density [s/mm^2]
0.00310
(f)

Fig. 2.3: Optimization for basic surface shapes with a Gaussian distribution as desired coating height. The coating error is coded in the surface color. The little balls show the gun positions, and their color indicate the spray time density.

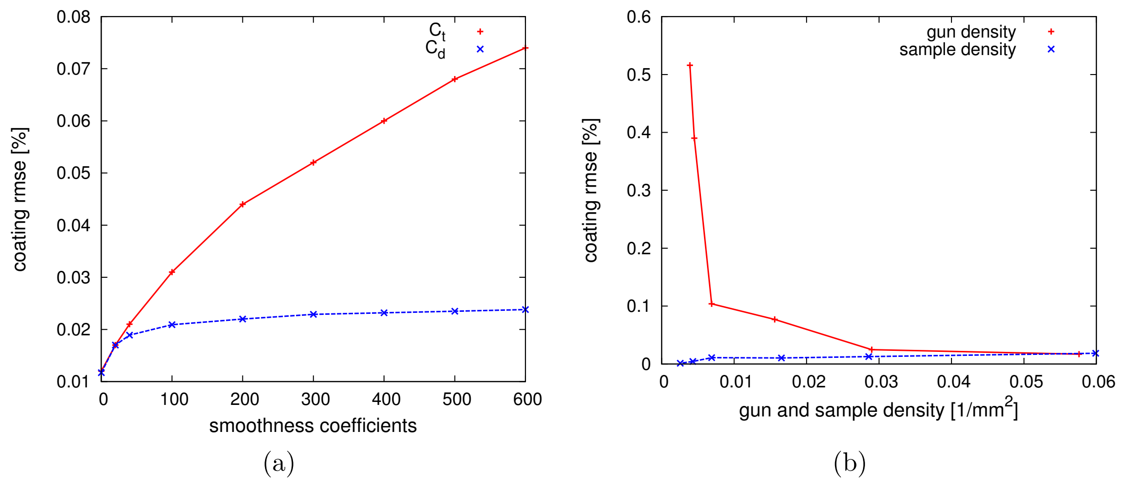


Fig. 2.4: The coating error (a) in dependence on the smoothness coefficients C_t and C_d, and (b) in dependence on the gun and sample point density.

distribution of the spray time density and the distance of the guns are of major concern for the evaluation. In all experiments, the opening angle of the spray cone has been set to $\gamma = 18.33°$, the volume flow rate factor to $V_T = 300\,\text{mm}^3/\text{s}$, the weights C_d and C_t of the objective function to 20, C_e to 1, and C_{sd} to 0.008.

In Table 2.1, the coating errors of the sprayed layer for the basic shapes are shown. The root mean square error has been calculated as the root of the squared difference between the sprayed layer height and the desired layer height, relative to the mean of the desired layer height. The desired layer height has the form of a Gaussian function plus a constant value. The density of the guns and the sample points is about the same for all examples. Figure 2.3 shows the optimization results for the basic surface shapes. The sphere and the cylinder patches are coated from the front and the back side. The relative layer height error, which is always below 0.115 %, is displayed by a color coding on the surface. The locations of the spray guns are shown by small spheres. Their colors indicate the individual spray time densities. It can be noticed that a low deposition error as well as a smooth variation of distances and spray time densities has been achieved, which indicates that the smoothness objectives and the density calculation are correct and work as expected.

Figure 2.4(a) shows the effect of the smoothness coefficients C_t and C_d on the coating error for the plane test surface. The coating error increases with an increase of the smoothness coefficients. The larger influence of the spray time density smoothness coefficient is related to the specific problem, where the gun distance is mostly uniform in the optimal solution.

Figure 2.4(b) shows the effect of the spray gun density and the sample point density on the coating error for the plane test surface. Unsurprisingly, the error decreases when the gun density is increased. When the gun density is too low, the coating error increases dramatically, because the spray cones do not overlap sufficiently anymore. On the other hand, the coating error is only slightly influenced by the sample point density.

Figure 2.5 shows results of timings of the Levenberg-Marquardt method. Flat rectangular surface patches with different surface areas have been used and there has been one point on $16\,\text{mm}^2$ in average for the gun base points as well as for the sampling points. For a surface area of $40,000\,\text{mm}^2$ this results in about 2500 gun base points and sampling points, respectively. The target coating is again based on a Gaussian function.

It can be easily seen that the computation time grows about linearly with the area of the surface. The time requirement of the visibility test, which is performed before the optimization, also increases linearly, and takes about 50 times less than the optimization process. The visibility calculation is executed before the optimization because the simple surfaces used here have no undercuts.

The Levenberg-Marquardt method converges after relatively few iterations. Figure 2.6 shows the values of the objective function depending on the number of iterations for a flat surface and a desired coating height in the form of a Gaussian function. After five iterations the optimization reaches an objective function value of about 10^{-6}, which is only very slightly improved in the following iterations.

Figure 2.7 shows a more complex surface of a real workpiece. The application background is to apply a coating on tools for deep drawing. The surface mesh consists of $24,680$ triangles. For the calculation, $7,537$ sample points and $3,097$ gun configurations have been used. The desired coating height has been 0.2 mm. The root mean square error achieved is 0.53 %, the maximum error 2.72 %, from initially 7.83 % and 18 %, respectively.

The distances of the spray guns have been constrained to the interval between 50 mm and 130 mm. The observed distances of the result lie between 50 mm and 129.5 mm, the mean distance is 92.5 mm. The corresponding footprint radius on the surface is approximately 30 mm. The observed spray time densities of the guns are between 0.00091958 s/mm^2 and 0.0012742 s/mm^2, the mean time is 0.0010088 s/mm^2. The overall time of calculation has been 160 s, of which 3.1 s have to be accounted to the visibility calculation. This example demonstrates the usefulness to optimize both the spray time and the spray distance, because the small geometric features (small radius) of the workpiece cannot be compensated by just the spray time. The guns have to move closer to the surface in order to shrink their area of influence.

2.9 Conclusions

An alternative scheme to the usual geometry-first strategy to path planning for production processes has been outlined. The so-called geometry-last approach starts with the calculation of an optimal cover of the workpiece with a finite set of tool configurations. This leads to the tool configuration cover problem which turns out to be of value for the geometry-first scheme as well. A formulation of the tool configuration cover problem as an optimization problem has been presented for the case of spray coating for which the approach is particular useful. A solution to the problem based on a descent method has been described. It has been shown in experiments that good solutions of the optimization problem can be obtained in a quite efficient way.

An issue of the definition of spray directions not considered in Section 2.5.4 is collision avoidance. The spray direction and distance interval have to provide a collision-free placement of the tool with respect to the workpiece. This could be achieved by sampling the spray directions and selecting a collision-free sample for each gun. The selected samples should be near the surface normal and provide a smooth variation along the surface, which leads to a new optimization problem. In this thesis, collision avoidance is postponed to the manipulator path optimization in Chapter 5.

The optimization of gun configuration covers has been performed over the gun distances and the spray time densities. A further parameter could be the spray direction which has been defined in advance up to now. A disadvantage is that the computation time may increase significantly and the additional impact on the quality of the results already achieved seems to be limited. For that reason it is postponed to future work.

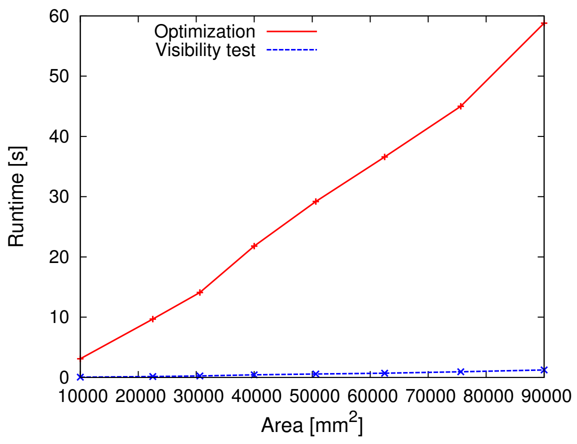


Fig. 2.5: Computation time for different surface areas on a flat plane.

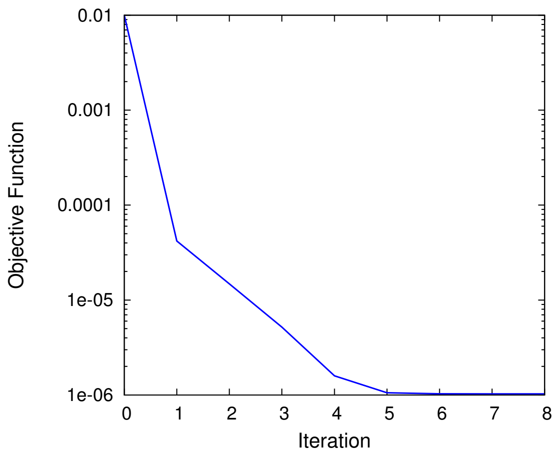
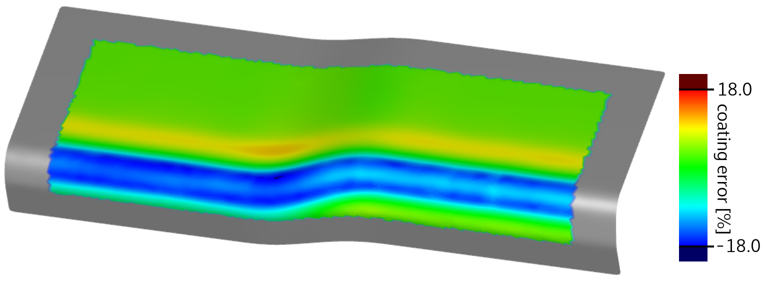
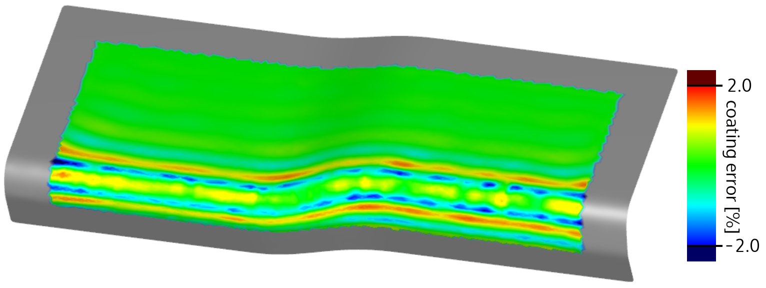


Fig. 2.6: Values of the objective functions reached on a flat surface with a Gaussian function as target coating.



(a) Initial coating error. rmse: 7.83 %, max: 18 %



(b) Optimization result. rmse: 0.53 %, max: 2.72 %

Fig. 2.7: Coating error for a complex surface. The coating error is coded in color on the surface.

The second step of the geometry-last approach is the construction of a path, taking the information of the first step into account. This information is of particular value in the case of free-form workpieces where paths adapted to the surface are required. This aspect is the topic of the next chapter.

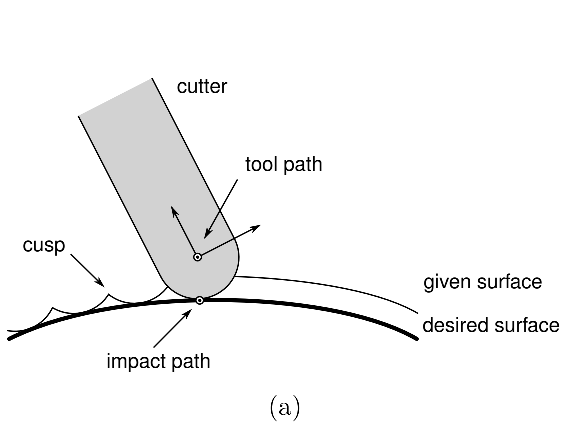
3 Tool-adaptive Offset Paths on Triangular Mesh Workpiece Surfaces

In practice, it is often preferred to use quasi-parallel offset curves as impact paths. They require to choose the offset so that a tool moving along the curves has the desired impact at every surface point. A problem with the constant offsets commonly used is that in cases where the region of influence of a tool is different across the surface, an offset value necessary in one region may lead to a curve offset lower than required in other regions. This chapter presents a general method of offset curve construction with tool-adaptive offsets. The offset path is obtained as a family of isolines of an anisotropic distance function relative to a seed curve on the workpiece surface. Anisotropy is defined by a metric tensor field on the surface. An application-independent algorithmic framework of the method for workpiece surfaces represented by a triangular mesh is presented. Its usefulness is demonstrated on the problem of varying cusp heights for milling and on the task of achieving a desired coating height for spray coating. The latter employs the path-geometry-last concept of Chapter 2 and presents an approach for its second step – the construction of a spray gun path.

3.1 Introduction

The central issue of path planning for path-oriented manufacturing processes is to design a tool path fulfilling the requirement of generating the desired workpiece. A *tool path* typically describes the motion of a tool by the motion of a frame with its origin at a tool center point, e.g. called cutter location (CL) path for milling. During its motion the tool interacts with the surface. The sequence of interaction points may be described by an *impact path*. For example, the impact path of a ball-end cutter, often called cutter contact (CC) path, is induced by the contact points of the cutter with the desired surface (Fig. 3.1(a)). For spray coating the curve induced by the intersection points of the centerline of the moving spray cone with the surface may be taken as impact path (Fig. 3.1(b)). In fact, the interaction of the spray cone with the surface is not restricted to a curve, but an impact path defined as curve allows to perform path planning on the desired surface, as does the impact path for milling.

Since the impact on the workpiece is the central issue, a natural approach to path planning is to start with the design of an impact path. Then a tool path which has the desired effect on the workpiece is derived relative to the impact path. The impact path has to be chosen so that several constraints and objectives can be satisfied and optimized, respectively. They concern kinematic properties of the tool path which should take into account constraints of the tool-surface interaction, like e.g. material removal or contact forces in the case of milling, or the volume flow rate of a spray gun. Furthermore, dynamic properties of the manipulator might be considered. Often such constraints are taken into account by qualitative heuristics which restrict the considered design domain to a potentially well-suited path type. The path-geometry-last approach of the preceding chapter increases the available information about the requirements of a production process, in this case spray coating, quantitatively



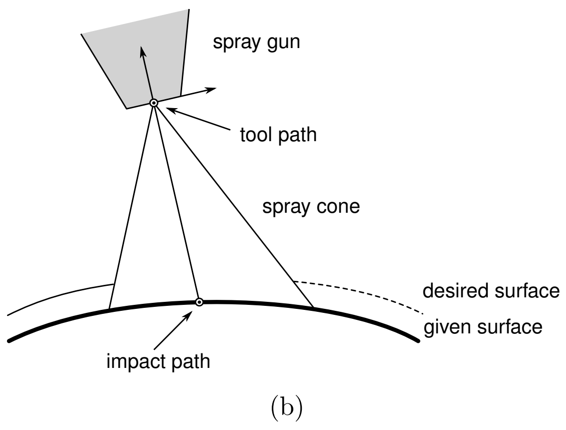
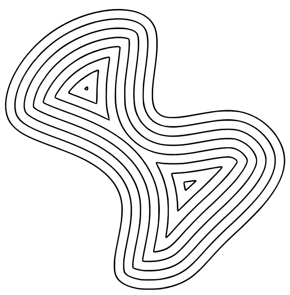


Fig. 3.1: Impact path and tool path of (a) a ball-end cutter and (b) a spray gun. The paths, indicated by $\odot$, are perpendicular to the image plane.


(a) Contour-parallel

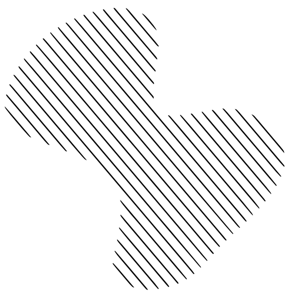

(b) Direction-parallel

Fig. 3.2: Path patterns.

with respect to the given specific planning task. This way, a better adapted and improved tool path may be achieved.

The approach of impact path-based path planning just described is restricted to those segments of a tool path which have impact on the workpiece. Sometimes additional tool path segments without impact are required which transport the tool to distant locations on the workpiece. An example is pocket machining by spiral-shaped milling tool path. A pocket is a local cavity of the workpiece. To reach or leave the center of the spiral a retraction of the tool is required which has no direct impact on the surface.

This chapter is concerned with the planning of impact paths under consideration of production process-related information. The task of impact path planning is to calculate a covering path on the surface such that it can be used for tool and manipulator path planning. Impact path topologies often used in practice are contour-parallel and direction-parallel paths (Fig. 3.2), and zigzag and spiral paths which may be derived from them. Those approaches of taking as uniform as possible distances in-between neighboring curve segments of an impact path lead to reasonable paths for many applications.

An approach to the construction of such curves on a surface is to consider isolines of a scalar function on the surface. An isoline, sometimes also called iso-curve or level set, is defined by all points on the surface which have the same scalar function value v_{iso}. A

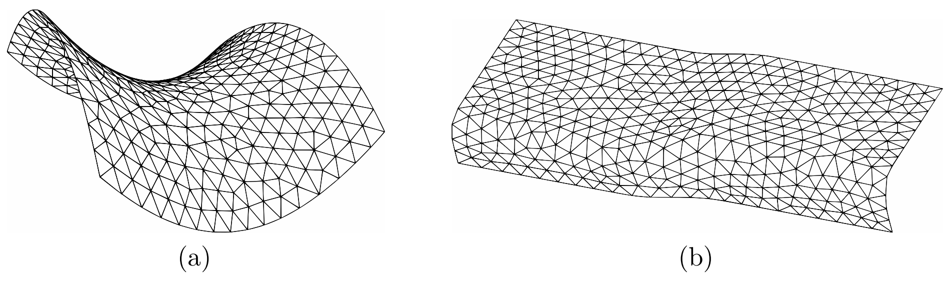


Fig. 3.3: The surfaces used for experimental evaluations, at a reduced mesh resolution for display purposes. (a) A saddle surface with 353 vertices and 639 triangles. (b) A workpiece surface with 396 vertices and 715 triangles.

family of nested curves is obtained by taking a sequence of values v_{iso} of equal difference. A particular advantage is the straightforwardness and stability by which such curves can be calculated.

Particularly interesting scalar functions are the distance functions. A distance function is induced by a source set on the surface. The scalar value of a point is the minimum (geodesic) distance of the point to the source set. A typical example of a source set is the boundary of a surface. This leads to the contour-parallel curve pattern. A property of distance functions making them particularly favorable is that the isolines have a mostly uniform geodesic distance, and thus are in some way parallel.

However, better results may be achieved by additional degrees of freedom emerging from locally adapting the inter-path-distances depending on the particular application. For example, it is common for spray coating to adapt the distance and speed of the spraying tool along a given path in order to achieve the desired coating height, cf. e.g. [35, 62]. An alternative could be to fix either speed or distance, or both, but vary the density of the path to get the desired coating height. The advantage could be a more uniform distance and speed which may be favorable for the manipulator moving the spray gun.

Adaptation of the inter-path-distance may be achieved by anisotropic distance functions defined by appropriate metric tensor fields on workpiece surfaces. Kim [58] describes such an approach for milling with respect to the cusp height. The goal is to achieve a uniform height of the cusps on the processed surface, independent from the curvature of the surface (Fig. 3.1(a)). This chapter will point out the general potential of the approach as a general method for other objectives and other production processes.

The first main contribution is an algorithmic framework of the general method on workpiece surfaces represented by triangular meshes [17] (cf. Fig. 3.3 and Section 2.5.3). The algorithmic framework can be implemented in a stable, reliable and efficient way at reasonable efforts and thus is well suited for practice. It uses an implicit path representation instead of a parametric representation as Kim does, and it takes into account several important achievements and results of mesh processing in recent years. Its basic idea is to define an anisotropic distance function on the mesh based on metric tensors assigned to its vertices. The metric tensors are defined specifically for the application process and they specify the necessary adaptation of the inter-path-distance. The impact paths are calculated as isolines of the anisotropic distance function by the marching triangle algorithm. If the resulting paths do not satisfy the requirements the metric tensors are adapted. This is iterated until the requirements are satisfied or no further improvement is achieved.

The application of the framework to the problem of uniform cusp heights leads to an algorithmically different realization of the approach by Kim [58], with the before-mentioned advantages. Figure 3.11(a) visualizes the cusp heights for an isotropic distance-parallel topology, i.e. with constant distances, while Fig. 3.11(b) shows a related anisotropic solution of locally varying path density.

The second main contribution is the application of the method to spray coating. Two novel approaches of path construction from the information provided by a spray gun cover are proposed, continuing the path-geometry-first approach of Chapter 2 by a solution of its second part, the path construction. They are based on a relation between gun velocity, path interval size, and spray time density, and a relation between the path interval and the coating height. As experimentally demonstrated, these heuristic models, although considerably simplifying, already lead to favorable metric tensors which may help to achieve a fast path calculation. Figure 3.16(a) shows the coating height errors for an isotropic direction-parallel path and Fig. 3.16(b) and 3.17 anisotropic improvements iteratively derived from it.

Section 3.2 gives a survey on related work. Section 3.3 provides the fundamentals of distance functions required in the chapter. Section 3.4 presents the method. It is divided in subsections corresponding to the steps of the method. Sections 3.5 and 3.6 are devoted to the demonstration of the method of path interval adaptation for milling and spray coating impact paths. Section 3.7 concludes the chapter.

3.2 Related Work

Parallel paths have found considerable interest in path-based production, in particular for technologies with tool contact like machining [67], but also for contact-free technologies like spray painting [24] and spray coating [35]. Other path types have been investigated, too, like space filling curves [67] and trochoids [37], but with less attention in research and application.

For the calculation of parallel curves, simple approximate methods of low accuracy, like e.g. parallel cutting planes, are in widespread use [67]. True parallel paths can be determined by mainly two approaches: explicit and implicit offsetting. Among those, explicit offsetting is often used, but it has the drawback of potential self-intersections [67]. Implicit techniques also exist for a long time. They have found increasing interest since the advent of the so-called level-set method [92] and in interesting applications of distance functions in computer graphics which stimulated the search for efficient data structures and algorithms for this purpose [54]. Among the applications is also path planning [12, 33, 60, 114], in particular for pocket machining. Implicit offset curves do not have self-intersections, but they still have sharp corners, which also occur in explicit offsetting. They have to be taken into account and, if they cannot be avoided in an application, have to be remedied.

Like for CAD, parametric surface representations of CAD models based on continuous mathematics have been preferred in path planning [67]. Recently, the surface representation by meshes has gained increasing interest. The use of meshes has formerly been mainly motivated by FEM-based calculations. Meanwhile, data sets obtained from 3D scanning techniques are the driving force for the immediate application of meshes for geometric modeling and CAD, and related intensive research [17]. Accordingly, meshes also have been used in path planning methods, but the development is still behind smooth representations [67].

Meshes are well suited for distance function representations and for its calculation. Mitchell et al. [80] have presented an algorithm for the calculation of distance functions on triangular mesh surfaces. This algorithm has later-on been improved in several ways [15, 96]. The method of the chapter makes use of this algorithm and extends it to anisotropic distance

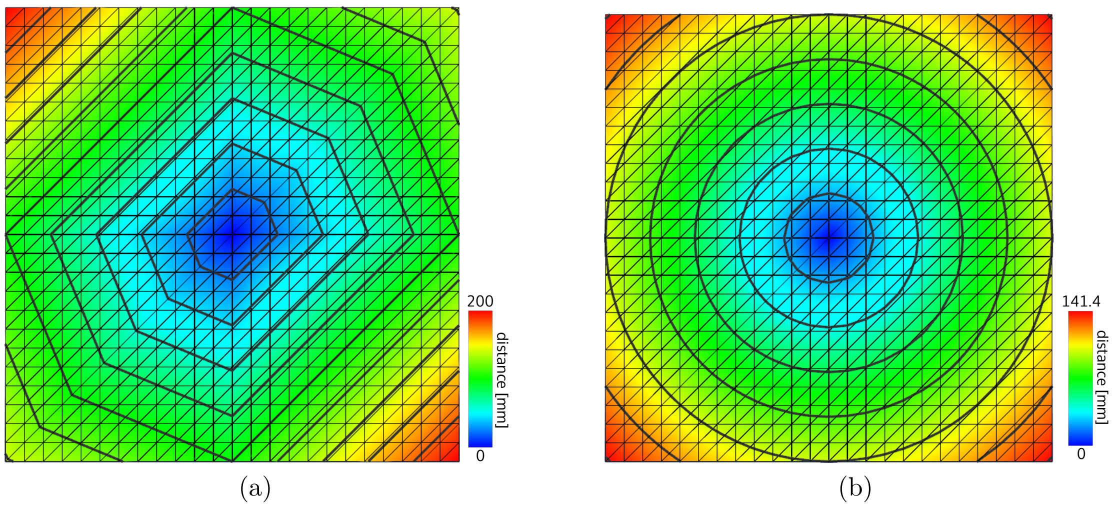


Fig. 3.4: A distance function on a regular mesh calculated by (a) the Dijkstra algorithm, and (b) the MMP algorithm. Notice the discretization artifacts arising from the regular structure in combination with the Dijkstra algorithm, which are also independent from the mesh density.

functions. Furthermore, efficient algorithms for the extraction of isolines from distance functions related to the marching cubes approach by Lorensen and Cline [77] are available and used in this chapter.

Another approach to distance function calculation is the fast marching algorithm on triangular meshes [61, 92, 93, 94]. It approximates the distance function at vertices by solving a discrete differential equation. The approximation quality depends on the mesh density and on the accuracy of the gradient estimation, which cannot be calculated exactly with just the vertex distance values.

An approach specialized to meshes is to consider the given mesh as a graph $G(M)$ induced by the vertices and edges of the mesh. If the seed set S is a subset of the mesh vertices, a distance function may be calculated for all mesh vertices $\mathbf{v}$ by determining the length of a shortest path in the $G(M)$ from $\mathbf{v}$ to S. The shortest paths can be determined by a suitable version of Dijkstra's shortest path algorithm [34]. However, a drawback of the graph-based approach is the occurrence of discretization artifacts. As an example, consider a regular triangular mesh in the plane, i.e. a rectangular grid with additional diagonal edges. Then the set of vertices of distance d from a fixed vertex does not approach the Euclidean circle of radius d even if d is big, as possibly could be expected at a first glance (cf. Fig. 3.4). Suggestions to improve this approach are based on the insertion of additional edges between nearby vertices or between new virtual vertices on the original edges [96].

Bounding the height of cusps is a classical objective of path planning for milling. Kim [58] gives a survey on the development of solutions on free-form surfaces. He also indicates that an implicit approach might be possible as an alternative to his explicit construction of anisotropic geodesic parallels by adaptive metric tensors. This is performed in the first case study of this chapter. Tchon et al. [98] give a further example of a useful application of anisotropic metrics in the different field of volume mesh generation. This underlines the importance of investigating possible further employments of the concept, as it is done in this chapter.

Related work of path planning for spray coating has already been compiled in Section 2.2. As noted there, efforts already exist for adaptive path construction which, however, have restricted power. The method presented in this chapter will provide an elegant solution of the second part of the path-geometry-last approach, the path construction, which has been left open in Chapter 2.

3.3 Distance Functions and Isolines

In this section the concepts of distance functions and isolines are recalled, with emphasis on triangular meshes. Their efficient algorithmic implementation on triangular meshes is also part of this section.

3.3.1 Distance Functions

The theoretic basis of distance functions on smooth surfaces is provided by differential geometry in form of the concept of Riemannian 2-manifolds [22]. A *Riemannian 2-manifold* is a surface augmented by several concepts. In particular, a Riemannian 2-manifold provides a metric d which allows to define a distance $d(\mathbf{p}, \mathbf{q})$ between two surface points $\mathbf{p}$ and $\mathbf{q}$ by the length of a shortest connecting path on the surface. In the following, a surface M with a metric d is denoted as a *metric surface* $\mathcal{M} = (M, d)$.

A triangular mesh is a special sort of metric surface. A canonical metric on a mesh is induced by the Euclidean metric in the triangles. For two points in one triangle, the metric coincides with the Euclidean distance. For two points in different triangles, polylines on the surface are considered. A polyline consists of a sequence of vertices consecutively connected by edges. The edges are straight lines in the triangles. Except possibly the first and last vertex, the vertices are located on edges or vertices of the mesh. The distance of the two points is the length of a shortest path of this type between them.

A *distance function* on a metric surface is defined relative to a source set, to which the distance is calculated, and stores the shortest distance to a source element at each element of the definition space. Formally, given a metric surface $\mathcal{M} = (M, d)$ and a source set $S \subset M$, the corresponding distance function d_S is defined as a real function on M with

$$d_S(\mathbf{m}) := d(\mathbf{m}, S) := \inf\{d(\mathbf{m}, \mathbf{s}) \mid \mathbf{s} \in S\}\,.$$

Figures 3.7(a) and (c) show examples of distance functions on free-form surfaces with a polyline as source, visualized by color coding.

3.3.2 Calculation of Distance Functions on Triangular Meshes

For triangle meshes and a source set S consisting of a single mesh vertex, Mitchell et al. [80] have presented an algorithm for the calculation of its distance function on a mesh, in the following called *MMP algorithm*. They also have described an extension to sets S of a finite number of vertices which can be easily adapted to other surface points which are not vertices. Several further improvements and extensions have been proposed by other authors. One useful modification of the MMP algorithm is due to Surazhsky et al. [96], who describe an efficient implementation of the original MMP algorithm with an additional approximation algorithm which merges adjacent windows if they are similar.

The MMP algorithm represents the distance function of S restricted to the edges of the mesh, in a way that the distance of a point inside a mesh triangle can be calculated from the information provided on the edges of the triangle.

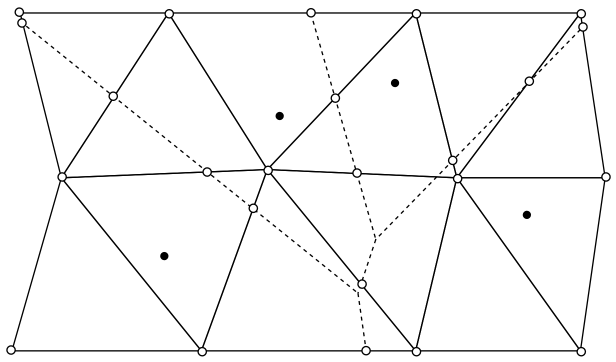


Fig. 3.5: Voronoi diagram (stippled lines) of four point sources (black points) on a triangular mesh. The Voronoi regions induce intervals on the edges of the mesh whose endpoints are indicated by small circles.

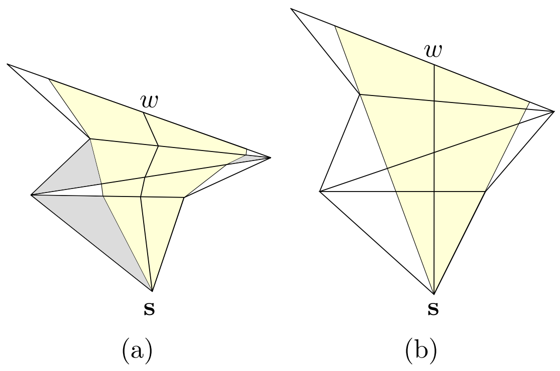


Fig. 3.6: Determining a shortest path on a spatial triangle mesh as a line segment on the locally unfolded mesh.

The representation of a distance function on an edge consists of intervals with disjoint interior, sometimes also called *windows* [15]. A window w represents a bundle of shortest paths from the source set to the points in the window, with the additional property that all paths emerge from the same source point $\mathbf{s}(w) \in S$, and the inner paths of a bundle do not traverse mesh vertices. This property implies that the windows are subsets of the intervals resulting from intersecting the mesh edges with the regions of the Voronoi diagram of S, cf. Fig. 3.5.

The *Voronoi diagram* of a finite set S of points is the set of all surface points having equal distance to at least two of the given points. It induces a partitioning of the surface into so-called *Voronoi regions* each of which consists of all surface points closer to one of the points in S than to any other one.

The windows and the related distance information are calculated in a breadth-first process similar to the algorithm of Dijkstra for shortest paths in graphs. The breadth-first process is controlled by a priority queue. The priority queue contains windows which are processed according to increasing distance from the source set.

Shortest paths which have just vertices in the interior of mesh edges can be calculated by the mechanism of unfolding of Fig. 3.6. Mesh unfolding converts a shortest path into a straight line in a plane mesh which results by rotating the triangles subsequently around these edges so that they become co-planar. Figure 3.6 illustrates this observation and depicts the example of a window in yellow. Hence, window propagation can be performed by locally unfolding the mesh.

Shortest paths including mesh vertices are treated specifically, not affecting the basic principle of the approach. It is known that for mesh vertices on shortest paths the sum of the angles of the incident triangles is at least 2π. Vertices with the sum of angles larger than 2π are called *hyperbolic vertices* since they correspond to that type of points of differential geometry of smooth surfaces [22]. Hyperbolic vertices are taken into account in the MMP algorithm by adding the hyperbolic vertices to the source set. This is done dynamically during execution of the algorithm as soon as a hyperbolic vertex is reached and its shortest distance to the current source set has been determined. Hyperbolic vertices are annotated with their shortest distance as a distance offset. During further processing, the distance offsets are added back to the shortest distances to the hyperbolic vertices.

Somewhat more detailed, the windows may be stored as 6-tuples $(b_0, b_1, d_0, d_1, \sigma, \tau)$, where b_0 and b_1 are the interval boundaries, d_0 and d_1 are the distance values at the interval boundaries, σ is the distance offset which is added to the distance values and τ is a boolean value deciding on which side of the edge the source point lies. The unfolded location $\mathbf{s}$ of the source point of the window can be calculated from d_0, d_1, and σ. σ is non-zero if the source point is one of the added hyperbolic vertices. During the algorithm, the windows are propagated along the neighboring triangle onto the other two edges. In the case that the propagated windows overlap with existing windows on an edge, then intersections between the overlapping windows are calculated so that the segments with larger distances can be removed.

Surazhsky et al. [96] describe an efficient implementation of the original MMP algorithm with an additional approximation algorithm which merges adjacent windows if they are similar. An efficient method to compute the shortest geodesics is also presented.

The original MMP algorithm has a worst-case time complexity of $\mathcal{O}(n^2 \log n)$, n the number of vertices. However, Surazhsky et al. [96] state that the average time requirement in practice is close to $\mathcal{O}(n^{1.5})$.

Distance functions of a source different from finite sets of points may be approximated by replacing the source by a finite set of sample points. For sources defined by polylines, Bommes and Kobbelt [15] have developed an extension of the MMP algorithm which may treat them immediately. Polyline approximations of curved sources, like e.g. a region contour, may need fewer elements than approximations by sample points, in particular in segments of low curvature. However, since the MMP agorithm can easily be modified to further sorts of distance calculations required in the following, the approximation of sources by a finite set of sample points is preferred. Figure 3.7(a) and (c) show two examples of distance functions calculated from point samples of curves.

3.3.3 Isolines on Triangular Meshes

Given a continuous scalar function φ on a surface M and a real number t, a *set of isolines* $\{I_t^j\}_j$ is a set of maximally connected sets of points $\mathbf{m}$ in M whose value $\varphi(\mathbf{m})$ is t, that is

$$\dot{\bigcup}_j I_t^j := \{\mathbf{m} \mid \varphi(\mathbf{m}) = t\}.$$

For distance functions in particular, an isoline represents a connected set with a constant distance to the source set. Figures 3.7(b) and (d) show examples of families of isolines for a sequence of equidistant iso-values.

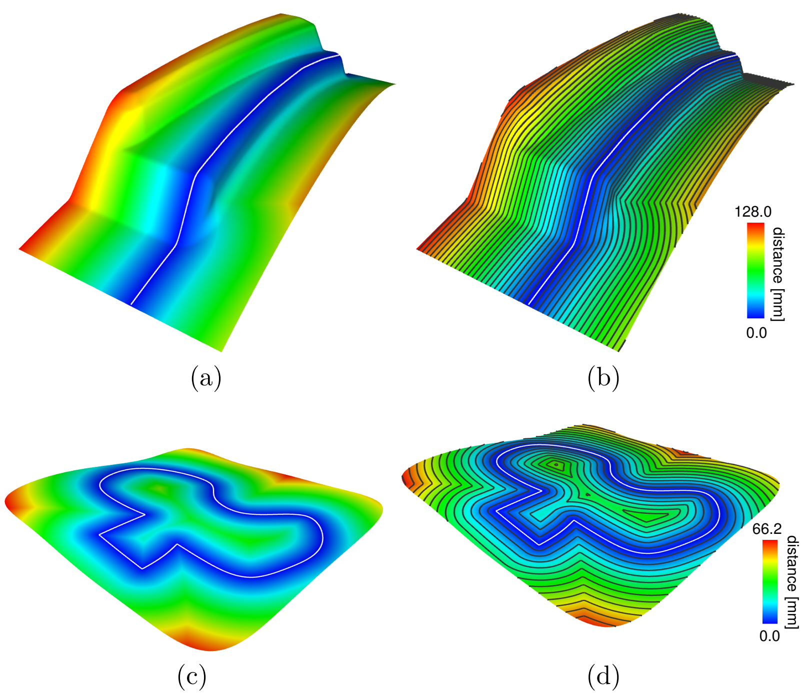


Fig. 3.7: The distance functions of a point-sampled approximation of (a) an open and (c) a closed curve (in white) on a surface represented by a triangular mesh, in color-coded visualization and calculated with the MMP-algorithm. Figures (b) and (d) show families of isolines induced by the distance functions.

Algorithm 2 Mesh-based implicit path adaptation by metric tensor fields (MTF method).

```
Input: A workpiece surface represented as a triangular mesh, an objective of a production
    process.
Output: A family of neighboring impact path segments which satisfies the objective.
 1: Define a seed set for the paths to be calculated.
 2: Define a process-dependent metric tensor field.
 3: Calculate an anisotropic distance function induced by the metric tensor field.
 4: Smooth the anisotropic distance functions (optional).
 5: Extract the paths as isolines of the distance function.
 6: If the result satisfies the requirements, or if the iteration did not achieve any further
    improvement, then stop. Otherwise, adapt the metric tensor field and iterate with
    step 3.
```

In the case of triangular surface meshes, approximate isolines can be easily extracted from a distance function with the marching triangles algorithm, analogously to the marching cubes algorithm [77], but with triangles instead of cubes. The quality of the intersection point calculation between isolines and edges in the marching triangles algorithm depends on the distance function algorithm. With the MMP algorithm and its extensions, exact intersections can be calculated which mostly results in well approximating isolines (Fig. 3.7(b), (d)).

Difficulties only may arise at non-differentiable regions of a distance function where isolines have a high curvature. There, the marching triangles algorithm may deliver bad results due to an unfavorable tessellation. The problem can be reduced by subdividing the mesh as presented by Bommes and Kobbelt [15].

For the application as impact paths, the resulting set of contours needs to be transferred into a single path. This can be achieved by grouping the contours in maximum stacks of "concentric" contours. The contours in every stack are transferred into one spiral in the contour-parallel case (cf. Fig. 3.10), or in one zigzag-line in the direction-parallel case. Then the resulting path segments are connected by non-impact tool paths in a minimizing way, leading to a generalized traveling salesman problem.

3.4 The MTF Method

The aim is to find a family of neighboring impact path segments which satisfies a given objective of the production process. The topology of the path segments can be direction-parallel or contour-parallel, but the distance between neighboring segments may vary.

The method proposed here for treating this problem is *mesh-based implicit path adaptation by metric tensor fields* (abbr. *MTF method*). Algorithm 2 gives a survey of its steps.

For the adaptation to a specific application problem, a definition of the metric tensor field in step 2 and its adaptation in step 6 based on a model of the production process, as well as a possibility to check the quality of the result in step 6, have to be provided. The model does not need to be exact. However, the better the model the more reliable and faster the calculation is.

The method can be used in a virtual as well as in a real setting. In a virtual setting the quality of the result is checked by simulation of the process. The advantage of the real experiments is that simulation errors are excluded. However, if many iterations are required, a pure real setting is usually not practical because of the time and costs required.

The following sections are devoted to the different steps of the MTF method, except of step 5, which has already been treated in Section 3.3.

3.4.1 Seed Sets of Isolines

The seed set of step 1 is responsible for the basic shape of the desired parallel curves. Each of the parallel curves is defined by all surface points having the same given distance to the seed set. The seed set is usually chosen as a curve. For the direction-parallel path topology the seed curve may be a curve which traverses the surface starting and ending at boundary points. The seed curve may be provided interactively by the user or automatically calculated according to desirable objectives. The seed curve should have a low curvature in order to avoid singularities of the distance function. Another objective might be a low number of parallel curves induced by the seed curve in order to keep the number of turns of the tool low. A possible heuristic is to choose an as long as possible seed curve.

For contour-parallel curves, using the contour as the seed curve is the natural choice. The contour-parallel topology is less suited for contours with locations of high curvature since those may cause singularities of the distance function.

For surfaces of complex structure, e.g. with several boundary loops, for both topologies a seed set may consist of several disjoint curve segments.

3.4.2 Metric Tensor Fields on Meshes

Metric tensor fields required in step 2 of the method are a concept of *differential geometry* [22]. Metric tensor fields allow to define distance concepts on surfaces which are different from the usual Euclidean distance. If the metric tensor field changes, the distances on the surface change as well, while the geometry of the surface remains untouched. For example, metric tensor fields may be used to emulate the metric distortion caused by surface warping on the original surface. A crucial observation is that parallel curves of distance 1 with respect to a distorting metric tensor field may have a varying distance when considered with respect to the Euclidean surface metric. The adaptation of paths will be achieved in such a way.

A *metric tensor field* is a function on a surface which assigns a metric tensor to every surface point [22]. A *metric tensor* $D_{\mathbf{p}}$ is a positive definite, symmetric bilinear form in the tangent space at a surface point $\mathbf{p}$, i.e. the space of all spatial vectors orthogonal to the surface normal at $\mathbf{p}$. The metric tensor $D_{\mathbf{p}}$ can be expressed as

$$D_{\mathbf{p}} := \begin{pmatrix} d_1^2 & 0 \\ 0 & d_2^2 \end{pmatrix}, \tag{3.1}$$

where d_1 and d_2 are the metric distortion or scaling factors along the first and second principal metric direction. The *principal metric directions* are two orthogonal tangent directions in which the metric distortion is maximum and minimum, respectively. The formula holds if the principal metric directions are taken as the basis of the tangent vector space, else an additional basis transformation has to be applied to the tensor. The length of an arbitrary tangent vector $\mathbf{t}$ at $\mathbf{p}$ under the distorted metric differs from its Euclidean length by a distortion factor which is given in the relation

$$\|\mathbf{t}\|_{D_{\mathbf{p}}}^2 := \mathbf{t}^\top D_{\mathbf{p}} \mathbf{t} = \|\mathbf{t}\|_2^2 \cdot (d_1^2 \cos^2\theta + d_2^2 \sin^2\theta),$$

of the square lengths. θ denotes the angle from the metric's first principal direction to $\mathbf{t}$ and $\|.\|_2$ is the Euclidean vector norm.

The classical theory just outlined does not apply immediately to triangular meshes because of missing sufficient differentiability. However, the transfer of the concepts of differential geometry to triangular meshes has been a topic of intensive recent research [14, 31, 49], and it is also possible to define coherent concepts of metric tensors on meshes.

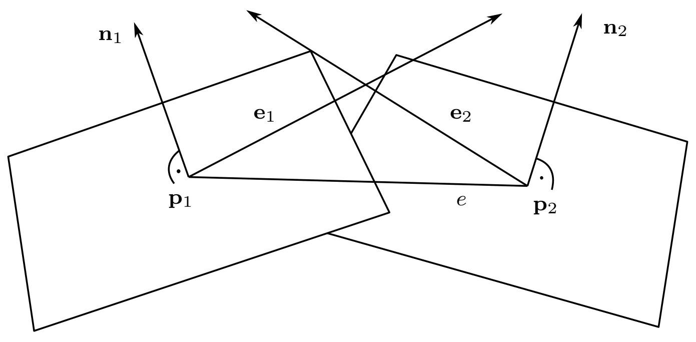


Fig. 3.8: Scaling of a mesh edge with respect to the metric tensors at its vertices.

The *canonical metric on a triangular mesh* is induced by the Euclidean metric in the triangles. For two points in one triangle, their distance is the Euclidean distance. For two points in different triangles, polygonal chains on the surface are considered. A polygonal chain consists of interior vertices located on edges or vertices of the mesh which are connected by straight lines in triangles. The distance of the two points is the length of a shortest chain of this type.

One possibility of changing the canonical metric of a triangular mesh is by assigning a new distorting metric tensor to every triangle which is used for every point of a triangle. Here, an alternative is pursued. It considers *metric tensor fields on triangular meshes* which are defined by assigning a metric tensor to every vertex. One reason is that from those vertex-related tensors compatible triangle tensors can be easily derived, as will be described in the following.

The definition of the metric vertex tensors uses a normal vector assigned to every mesh vertex. A simple possibility to get a vertex normal is a normalized, weighted sum of the normals of the incident triangles of the vertex. The plane perpendicular to the normal is considered as tangential plane, and the space of all vectors orthogonal to the normal define the tangential space. The metric tensor of a vertex is defined over its tangential space.

Given the vertex tensors, a piecewise constant metric tensor field on the triangular mesh is obtained by edge scaling. For this purpose, the edge e is orthogonally projected onto the tangent planes of its two vertices (Fig. 3.8). The lengths of the resulting line segments, considered as tangent vectors $\mathbf{e}_1$ and $\mathbf{e}_2$, are calculated with respect to the metric tensors of the two vertices according to Eq. (3.1). The scaled length of e is the mean value of the two results.

For reasonably defined vertex tensors, the scaled edges of a triangle induce a triangle, too. Then the original triangle is mapped one-to-one on the new triangle using barycentric coordinates. The resulting metric deformation can be described by a constant tensor in the original triangle. The tensors defined this way represent the desired piecewise constant tensor field.

The distance of two points according to the resulting tensor field is defined as follows. The new distance of two points in one triangle is the distance of their images in the deformed triangle. The new length of a polygonal chain is obtained as the length of its image on the deformed mesh. The distance between two points in different triangles is the minimum length of interconnecting polygonal chains of the images of the two points on the deformed mesh.

The time requirement of step 2 depends on the complexity of the tensor calculation. The latter is application-specific. The number of tensors is equal to the number of mesh vertices.

3.4.3 Anisotropic Distance Function Calculation on Meshes

Step 3 of the method concerns the calculation of an anisotropic distance function based on a given metric tensor field. Given a seed set S on a metric surface, the *distance function* is a function on the surface which gives the distance of any surface point $\mathbf{p}$ to S (Section 3.3). If $\mathbf{p}$ is in S, then the distance is 0. Otherwise, the distance of $\mathbf{p}$ is the infimum of all distances of $\mathbf{p}$ to any point of S. If the metric is defined by a metric tensor field with tensors with different scaling factors, the distance function is called *anisotropic*. If all scaling factors of all tensors are identical, as is the case for the canonical metric, the distance function is *isotropic*.

The MMP algorithm (Section 3.3.2) can be easily extended to anisotropic distance functions by locally applying the edge scaling described in Section 3.4.2 during the execution of the algorithm with the usual metric. This means that the algorithm remains basically unchanged, but is executed on the virtual mesh obtained by local deformation.

The quality of the resulting piecewise linear distance function is sensitive to the mesh tessellation. In order to cope with this problem, the algorithm of Bommes and Kobbelt [15] includes an adaptive refinement scheme based on mesh subdivision to increase the accuracy. For anisotropic distance functions the variance of the metric tensors has to be additionally taken into account. This is achieved in a preprocessing step by refining the mesh by adaptively subdividing edges by edge splits according to the same topological scheme. The metric tensor of a split point is interpolated from the metric tensor of the vertices of the split edge. A criterion for subdividing an edge is the size of the scaling factors resulting from the metric tensors at the two edge vertices. If the absolute difference exceeds a given threshold, then the edge is split in the middle and the criterion is evaluated on the new edges.

Tensor interpolation may be performed in a piecewise linear manner by first rotating the tensors at the vertices into a common coordinate system, which could be e.g. formed by an edge as x-axis and the face normal as z-axis. Then the resulting tensors are interpolated by a linear component-wise interpolation. Further advanced tensor field interpolation methods like the eigenvector-based interpolation of Hotz et al. [50] could also be used.

The examples presented in this chapter are calculated with this extension of the algorithm of Bommes and Kobbelt [15].

3.4.4 Smoothing of Real-valued Surface Functions

The purpose of distance function smoothing in step 4 is to remove sharp corners of the paths arising at singularities of the distance function (cf. Fig. 3.9). Singularities may occur at surface points for which the closest point in the seed set is not unique. At such points the distance function usually is just continuous but not differentiable. The smoothing step is optional since it is only required if sharp corners have to be removed.

A possibility of *smoothing of distance functions* is by a smoothing filter. A classical smoothing approach is convolution. For one-dimensional real functions f and g the convolution is defined as

$$h(x) = f * g(x) = \int_{-\infty}^{\infty} f(\xi) \cdot g(x-\xi) \mathrm{d}\xi \, .$$

From the calculus of differentiation and integration it is known that h is differentiable if g is differentiable, even if f is not [20]. By discretizing this concept to real functions on triangular meshes (cf. [17] for the Laplace-operator as an example), the distance function can be smoothed by replacing the function value $f(\mathbf{v}_i)$ of a vertex by a normalized weighted

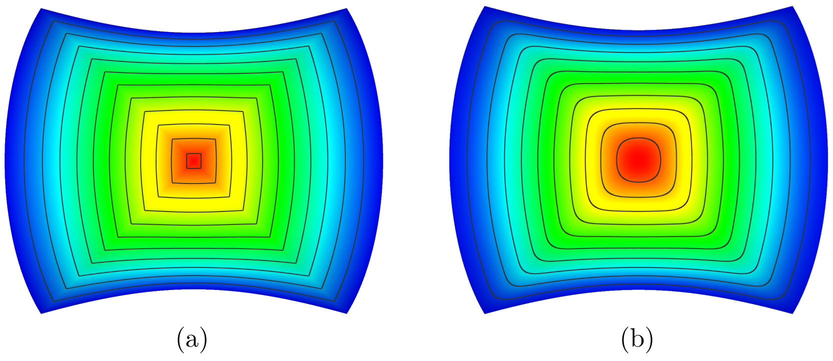


Fig. 3.9: Smoothing of isolines by distance function smoothing. (a) Original distance function with singularities. (b) Smoothed distance function. The seed line is the contour of the surface. The color indicates the distance function. Points colored in blue are close to the seed line, whereas points of red color are the furthest away.

average

$$\overline{f}(\mathbf{v}_i) = \frac{1}{\sum_{j \in R(i)} g(\|\mathbf{v}_j - \mathbf{v}_i\|) \cdot A_j} \sum_{j \in R(i)} f(\mathbf{v}_j) \cdot g(\|\mathbf{v}_j - \mathbf{v}_i\|) \cdot A_j \ ,$$

of the function values $f(\mathbf{v}_j)$ in an environment $R(i)$. A_j is the area of the Voronoi region of a vertex j (cf. Eq. 2.12). Weighting by the Voronoi area is caused by a discretization of the integral of convolution by using piecewise constant functions in the Voronoi regions. The environment $R(i)$ includes all vertices of geodesic distance to $\mathbf{v}_i$ less than a given radius $r > 0$. A canonical type of differentiable filter function is the Gaussian function $g(d) = 1/(2\pi\sigma) \cdot \exp(-d^2/(2\sigma^2))$ with $\sigma = r/2$.

3.5 Cusp-height Adaptation in Milling

The surface resulting from moving a milling tool along parallel path segments consists of parallel grooves separated by ridges (Fig. 3.1(a)). The aim of cusp height adaptation is to modify the path segments so that the cusps of the ridges do not exceed a given bound. This section employs the MTF method, without iteration. For this purpose, according to Section 3.4 a metric tensor field for step 2 of the method has to be designed so that the resulting isolines, taken as impact path (cutter contact path), fulfill the requirements.

The metric tensor field defined in the following takes into account the curvature of the workpiece surface and the radius of the tool ball. For the quantitative characterization of the curvature of a smooth surface differential geometry offers the concept of curvature tensors.

3.5.1 Curvature Tensor on Triangular Meshes

Like the metric tensor, the *curvature tensor* $K_\mathbf{p}$ of a smooth surface is also a positive definite, symmetric bilinear form. It describes the curvature behavior at a surface point $\mathbf{p}$. For every tangent vector $\mathbf{t}$ at $\mathbf{p}$, $K_\mathbf{p}$ calculates the curvature of the intersection curve of the surface with the plane spanned by $\mathbf{t}$ and the normal at $\mathbf{p}$, scaled with the square length of $\mathbf{t}$. In general, two orthogonal tangent directions exist at which the curvature becomes extreme. These directions are called *principal directions*, and the corresponding curvature values κ_1 and κ_2 are denoted as *principal curvatures.* If the normalized principal curvature directions are taken as the basis of the tangent space then the curvature tensor can be expressed as

$$K_{\mathbf{p}} := \begin{pmatrix} \kappa_1 & 0 \\ 0 & \kappa_2 \end{pmatrix}.$$

Analogously to the metric tensor, the curvature in a normalized tangent direction $\mathbf{t}$ at $\mathbf{p}$ can be calculated with the curvature tensor $K_{\mathbf{p}}$:

$$K_{\mathbf{p}}(\mathbf{t}) = \mathbf{t}^{\top} K_{\mathbf{p}} \mathbf{t} = \kappa_1 \cos^2\theta + \kappa_2 \sin^2\theta\,, \tag{3.2}$$

with θ being the angle from the first principal curvature direction to $\mathbf{t}$ in the tangent plane.

Like for metric tensors, the concept of curvature tensors cannot be immediately applied to triangular meshes because of the lack of differentiability. A possibility to cope with this problem is by reduction to the continuous case by local surface fitting using e.g. moving least squares [70]. A sophisticated method of direct calculation of a *curvature tensor on triangular meshes* has been presented by Alliez et al. [1]. Like in the continuous case, the curvature tensor is described by two orthogonal directions of principal curvature in the tangent space, and two values of principal curvature for these directions. A *curvature tensor field on a triangular mesh* assigns a curvature tensor to every vertex over its tangential space.

The computation time of the curvature tensor of a vertex according to Alliez et al. [1] is proportional to the number of vertices in its neighborhood involved in its computation. This number can be considered as constant in the number of mesh vertices.

3.5.2 Metric Tensor for Cusp Height Adaptation

The quality of approximation of the desired surface by a ball-end cutter depends on the local curvature of the surface. Lin and Koren [72] have proposed a path interval bound

$$\omega_0 = \sqrt{\frac{8h_0}{\kappa_t - \kappa}} \tag{3.3}$$

which limits the cusp heights. κ_t is the tool curvature, κ the surface curvature orthogonal to the path, and h_0 the desired maximal cusp height of a tool path. This definition is based on an approximation of the tool and the surface orthogonal to the path by circles, making it easier to calculate the remaining material after machining.

Equation (3.3) allows to construct a metric tensor which leads to parallel path segments fulfilling this property. The desired metric tensor is obtained by taking the normalized principal curvature directions as principal metric directions and using the scaling factors

$$d_i := \frac{1}{\omega_{0,i}} = \sqrt{\frac{\kappa_t - \kappa_i}{8h_0}},\ \ i = 1, 2\,. \tag{3.4}$$

This results in the metric tensor

$$D_{\mathbf{p}} = \begin{pmatrix} d_1^2 & 0 \\ 0 & d_2^2 \end{pmatrix} = \begin{pmatrix} \frac{\kappa_t - \kappa_1}{8h_0} & 0 \\ 0 & \frac{\kappa_t - \kappa_2}{8h_0} \end{pmatrix}. \tag{3.5}$$

Applied to a surface tangent vector $\mathbf{t}$ orthogonal to the path, and using Eq. (3.1) and (3.2), this metric results in a scaling by $1/\omega_0$:

$$\begin{aligned}\|\mathbf{t}\|^2_{D_\mathbf{p}} &= \mathbf{t}^\top D_\mathbf{p}\mathbf{t} = \|\mathbf{t}\|_2^2 \cdot (d_1^2\cos^2\theta + d_2^2\sin^2\theta)\,,\\ \|\mathbf{t}\|_2^2 &= \frac{\|\mathbf{t}\|^2_{D_\mathbf{p}}}{d_1^2\cos^2\theta + d_2^2\sin^2\theta}\\ &= \|\mathbf{t}\|^2_{D_\mathbf{p}} \cdot \frac{8h_0}{\kappa_t - (\kappa_1\cos^2\theta + \kappa_2\sin^2\theta)}\\ &= \|\mathbf{t}\|^2_{D_\mathbf{p}} \cdot \frac{8h_0}{\kappa_t - \kappa} = \|\mathbf{t}\|^2_{D_\mathbf{p}} \cdot \omega_0^2\,.\end{aligned}$$

Hence, an iso-value increment of 1 for the generation of the isolines in step 5 of the method leads to path intervals between neighboring curve segments satisfying Eq. (3.3). Thus, a cusp height of at most h_0 is reached.

This way an implicit version of the parametric approach by Kim [58] is achieved. In contrast to the approach by Kim, the metric tensor (3.5) is immediately specified in the tangent space of the surface, not in the parameter space. However, it is compatible with the metric tensor $\mathbf{H}$ derived by Kim. This can be concluded by considering local quadratic approximations of the workpiece surface, corresponding to the curvature tensor. These approximations have the type of the example of Section 6.1 of Kim's work. The tensor $\mathbf{H}$ of this example coincides with the metric tensor (3.5), if the different iso-value increment value of $2\sqrt{h_0}$ in Section 4.1 of Kim's work is taken into account.

The concept of metric tensors by Kim includes arbitrary cutter shapes as long as the surface of the rotating cutter is sufficiently differentiable. In that case, the approach of local quadratic approximation also allows the transfer of the related metric tensors to the MTF method of this chapter. For 3-axis machining the path adaptation is immediately possible. Kim [58] mentions that the tensor concept can also be extended to 5-axis machining using other types of smooth tools if tool orientations are preassigned over a part surface. However, for non-smooth tools, or tool orientations depending on the path direction, further research is required. An approach to derive appropriate tensors could be to consider the silhouette of the rotating tool in its preassigned orientation, in view direction tangential to the path. For example, the bottom of the silhouette of an inclined cylindrical, flat-bottom end-mill has a smooth elliptic shape. The silhouette shape depends on the orientation of the tool. The idea is now to use the silhouette shapes in all tangential directions at a surface point in order to derive relations to the desired maximal cusp height like the one of Eq. (3.3) for the circular silhouette shape of a ball-end cutter. Those relations could again be employed to create a metric tensor.

3.5.3 Evaluation

The cusp-height adaptation has been evaluated on a saddle surface and on a mechanical engineering part, cf. Fig. 3.3, 3.10 and 3.11. The milling is simulated by point-sampling the path and calculating the ray-sphere intersection for each mesh vertex along its normal with the ball-end tool placed at every sampling points. The resulting heights, linearly interpolated over the mesh, are visualized by color. The width of the surfaces is 200 mm and 350 mm, respectively. The tool radius is 25 mm for the saddle surface and 12 mm for the mechanical engineering part.

The saddle surface offers different types of curvature and thus shows a region-dependent cusp height for constant path intervals, cf. Fig. 3.10(a). The distance function seed is in the center of the saddle surface. A spiral tool path is created by interpolating consecutive concentric isolines. The tool path follows the blue curve, where the cusp height is zero. The maxima of the cusps in-between are colored in red. After the cusp-height adaptation the cusp-height is distributed uniformly over the surface, as can be noticed from the uniform, thin red regions in Fig. 3.10(b). The mechanical engineering part contains a region with a high curvature, where the cusps for constant path intervals are much higher than on the rest of the surface, cf. Fig. 3.11(a). The distance function seed is at the lower boundary curve of the mechanical engineering part. The adapted path lowers the cusp height in that region while leaving it unchanged in the more planar regions of the surface, cf. Fig. 3.11(b).

The runtime of the distance function calculation is slightly more than 3 s for the saddle surface mesh with 359040 triangles, and slightly less than 3 s for the mesh of the mechanical engineering part with 272736 triangles, on an Intel Core i7-2600 processor at 3.4 GHz.

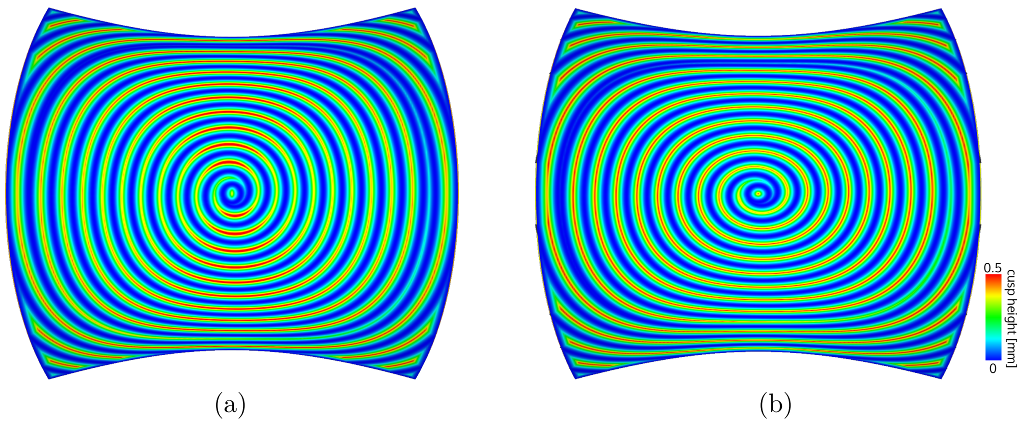


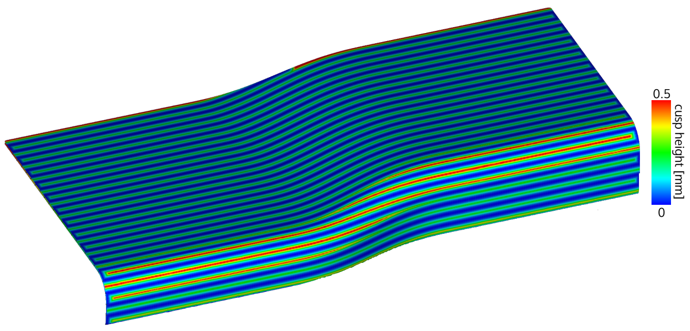
Fig. 3.10: Visualization of the cusp height induced by a path for (a) an isotropic and (b) a curvature-adapted anisotropic distance function on a saddle surface. As can be noticed from the non-uniform distribution of the red color, the cusp error is depending on the curvature in the isotropic case, while the thin uniform red regions in the anisotropic case indicate that the cusp error is uniform over the surface.

3.6 Path Interval Adaptation for Spray Coating

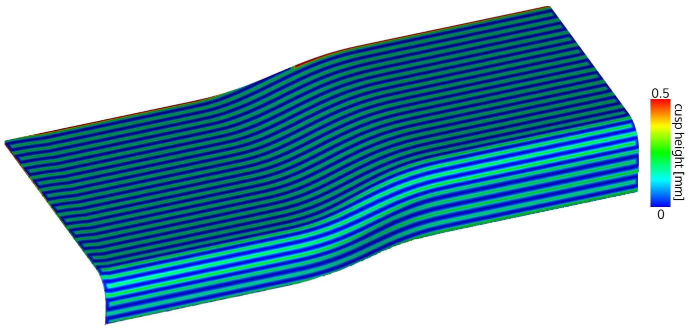
In the following, two approaches to the construction of paths to reach a desired coating height are presented. The first one is related to Chapter 2 and describes two possibilities to transfer its result to paths. The second one starts with any spray path and iteratively improves it by adapting the path intervals controlled by the coating height error. Both approaches use the MTF method.

3.6.1 Path Construction for Pathless Tool Pose Optimization

Chapter 2 describes methods to determine values of the gun pose parameters to achieve a desired coating height on a surface. One method is to distribute a finite set of spray gun poses over the surface, without any specific path, and to determine the distance and spray time of the poses by solving an optimization problem. An issue not treated in Chapter 2 is to transfer the path-independent solution to a spray gun path. In the following, two approaches to a solution of this problem are presented, velocity adaptation and path interval



(a)



(b)

Fig. 3.11: Visualization of the cusp height for a mechanical engineering part. For an isotropic distance function, the cusps are higher in regions of high curvature (a), which is compensated by the anisotropic distance function (b). Flat regions are not affected by the anisotropy.

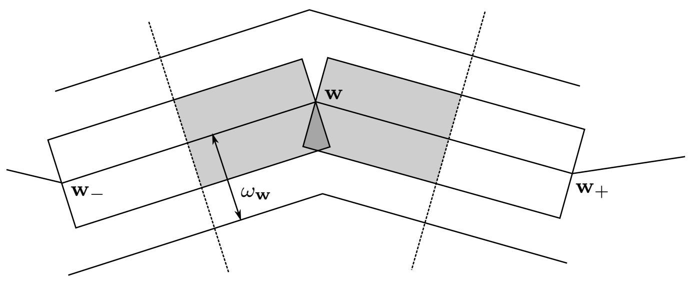


Fig. 3.12: Calculation of the spray gun density at a path vertex $\mathbf{w}$. The gray area represents an estimation of the Voronoi area around $\mathbf{w}$.

adaptation. Path interval adaptation uses the MTF method and provides a suitable metric tensor field for this problem.

The input of both approaches is a set of guns, one at every vertex of the given surface mesh. Like in Eq. 2.3, the location of the gun at vertex $\mathbf{v}$ is described by a vector $d_\mathbf{v} \cdot \mathbf{a_v}$ from vertex $\mathbf{v}$ to the gun reference point $\mathbf{q_v}$. Furthermore, a spray time density $t_{M,\mathbf{v}}$, a spray time $t_\mathbf{v}$, and a gun sampling density $\rho_\mathbf{v}$ with the relation

$$t_{M,\mathbf{v}} = t_\mathbf{v} \cdot \rho_\mathbf{v} \tag{3.6}$$

are given at every vertex $\mathbf{v}$ (cf. Eq. 2.9 in Section 2.5).

A spray gun path is represented over a polygonal impact path on the mesh surface which is given by isolines. The path vertices induce a further discrete gun covering. The gun pose at a path vertex $\mathbf{w}$ is determined by interpolation of the gun poses of the three vertices of the triangle in which $\mathbf{w}$ lies. Accordingly, the gun location vector $\hat{\mathbf{a}}_\mathbf{w}$ and distance $\hat{d}_\mathbf{w}$ at $\mathbf{w}$ are obtained by barycentric interpolation of the location vectors and distances at the mesh vertices.

The surface element $M_\mathbf{w}$ surrounding a path vertex $\mathbf{w}$ (cf. Section 2.5) in the new discrete gun cover is defined by the product of the path interval $\omega_\mathbf{w}$ and the mean length $l_\mathbf{w}$ to the neighboring vertices $\mathbf{w}_-$ and $\mathbf{w}_+$ along the path (cf. Fig. 3.12). It leads to the spray gun density

$$\hat{\rho}_\mathbf{w} := \frac{1}{l_\mathbf{w} \cdot \omega_\mathbf{w}} = \frac{2}{(\|\mathbf{w}_+ - \mathbf{w}\| + \|\mathbf{w} - \mathbf{w}_-\|) \cdot \omega_\mathbf{w}}$$

of the new gun covering.

The spray time density $\hat{t}_{M,\mathbf{w}}$ is obtained by barycentric interpolation from the spray time densities of the vertices of the triangle in which $\mathbf{w}$ lies. According to Eq. 2.9, the corresponding spray time is

$$\hat{t}_\mathbf{w} = \frac{\hat{t}_{M,\mathbf{w}}}{\hat{\rho}_\mathbf{w}} \,.$$

Then the velocity $v_\mathbf{w}$ on the path at $\mathbf{w}$ is the length of the path segment represented by $\mathbf{w}$ divided by the spray time assigned to the path vertex:

$$v_\mathbf{w} = \frac{l_\mathbf{w}}{\hat{t}_\mathbf{w}} = \frac{l_\mathbf{w} \cdot \hat{\rho}_\mathbf{w}}{\hat{t}_{M,\mathbf{w}}} = \frac{1}{\omega_\mathbf{w} \cdot \hat{t}_{M,\mathbf{w}}} \,. \tag{3.7}$$

This equation can be rewritten to

$$v_\mathbf{w} \cdot \omega_\mathbf{w} \cdot \hat{t}_{M,\mathbf{w}} = 1 \,. \tag{3.8}$$

In the special case of a mesh vertex $\mathbf{v}$ as path vertex,

$$v_{\mathbf{v}} \cdot \omega_{\mathbf{v}} \cdot \hat{t}_{M,\mathbf{v}} = 1 \text{ with } \hat{t}_{M,\mathbf{v}} = t_{M,\mathbf{v}}\,, \tag{3.9}$$

where $v_{\mathbf{v}}$, $\omega_{\mathbf{v}}$ and $\hat{t}_{M,\mathbf{v}}$ are the velocity, the path interval width and the spray time density of any spray path through mesh vertex $\mathbf{v}$.

Based on this, the approach of *spray path calculation by velocity adaptation* works as follows. It starts by choosing a family of approximate isolines, defined by a seed set and a monotonous sequence of iso-values of equal increments, as path on the mesh surface. Then the tool location and the spray time density are transferred to its vertices as just described. Finally the gun speed at every vertex is calculated from Eq. (3.8) as

$$v_{\mathbf{w}} = \frac{1}{\omega_{\mathbf{w}} \cdot \hat{t}_{M,\mathbf{w}}}\,,$$

where $\omega_{\mathbf{w}}$ is the path interval of the given path at vertex $\mathbf{w}$, i.e. the distance of the isolines. All this requires a constant computation time per path vertex, so that the overall asymptotic calculation time is proportional to the number of path vertices.

The approach of *spray path calculation by path interval adaptation* adjusts the interval between neighboring path segments subject to desired spray gun velocities along the path, by employing the MTF method, without iteration (section 3.4). This requires the specification of an appropriate tensor for step 2. For its definition, a scalar gun velocity field on the surface mesh has to be provided by assigning a velocity $v_{\mathbf{v}}$ to every mesh vertex $\mathbf{v}$. Using $\hat{t}_{M,\mathbf{v}} = t_{M,\mathbf{v}}$ and resolving Eq. (3.9) for $\omega_{\mathbf{v}}$ gives

$$\omega_{\mathbf{v}} = \frac{1}{v_{\mathbf{v}} \cdot t_{M,\mathbf{v}}}\,.$$

$\omega_{\mathbf{v}}$ is used to define the required metric tensor at vertex $\mathbf{v}$:

$$D_{\mathbf{v}} = \begin{pmatrix} \frac{1}{\omega_{\mathbf{v}}^2} & 0 \\ 0 & \frac{1}{\omega_{\mathbf{v}}^2} \end{pmatrix}. \tag{3.10}$$

The calculation of the tensor of a vertex is possible in constant time.

The necessary field of gun velocities can be defined heuristically. For example, it can be chosen according to possible constraints of the gun-moving manipulator, as follows. An isoline path, represented by a polygonal chain, is calculated for the given seed set and iso-value increment 1 for an isotropic metric tensor field with some scaling factor d. A scalar velocity value $v_{\mathbf{w}}$ is assigned to every path vertex $\mathbf{w}$ within the range feasible for the gun manipulator. The velocity values are transferred to the mesh vertices by interpolating the velocity value $v_{\mathbf{v}}$ of a mesh vertex $\mathbf{v}$ from the velocities at the path vertices close to $\mathbf{v}$. The possibility of taking manipulator constraints into account in this way is a particular advantage of path calculation by path interval adaptation over the first approach.

An iteration is not intended. However, if the result should not be satisfactory, an iteration according to the approach of the next section may be employed.

3.6.2 Coating-error-controlled Path Interval Adaptation

An alternative approach to adapting the interval between neighboring path segments based on the precalculated spray time density from the pathless optimization is to directly use the coating error of a given spray path. The advantage of coating-error-controlled path interval adaptation is that just the coating height for a given path has to be compared to the desired coating height; no optimization problem has to be solved in advance. The coating height can be determined by a deposition simulation based on a deposition model, or from a real experiment without the need of a deposition model. Although an initial spray path is required, in contrast to just an impact path on the surface mesh, a reasonable initialization of a spray path can be expected in many cases to be less computationally expensive than a complete initial pathless optimization.

The approach is also implemented by the MTF method. According to Chapter 3.4, this requires the definition of a tensor field (step 2), a possibility of quality checking (step 6), and an updating mechanism for the metric tensor field (step 6).

For the specification of the metric tensor field of step 2, certain path-independent parameter values have to be provided at the mesh vertices $\mathbf{v}$. The parameters are a gun position vector $\mathbf{r}_\mathbf{v}$, a scalar velocity value $v_\mathbf{v}$, and a path interval size $\omega_\mathbf{v}$. The values may be chosen heuristically, again possibly along desired constraints or objectives. From these parameter values, the metric tensor field is calculated according to Eq. (3.10).

For the quality check of step 6 of the method a spray gun path related to the resulting impact path is derived by displacing the impact path with the gun position vectors interpolated to the path vertices. Then the coating height $h_\mathbf{v}$ at every mesh vertex $\mathbf{v}$ caused by an execution of the initial spray path is determined e.g. by a simulation [111] or by a real experiment. The resulting coating height is checked for deviation from the desired coating height. If it is satisfactory, the algorithm stops.

Updating of the metric tensor (3.10) in step 6 is done by changing the path interval size at the mesh vertices to

$$\omega_\mathbf{v}^s = \omega_\mathbf{v} \cdot \overline{\left(\frac{h_\mathbf{v}}{h_\mathbf{v}^s}\right)}, \tag{3.11}$$

where $h_\mathbf{v}^s$ is the desired coating height at mesh vertex $\mathbf{v}$, and the overbar denotes a smoothing operator according to Section 3.4.4.

The asymptotic calculation time requirement of one iteration, apart from the simulation time, is the same as for the spray gun calculation by path interval adaptation of the preceding section.

Obviously, the path interval width in Eq. (3.11), without smoothing, remains unchanged if the desired coating height has already been achieved, and it increases or decreases if the coating height caused by the initial path is higher or lower, respectively, than the desired height. The smoothing operator is not mandatory, but it may be used to control the smoothness of the resulting path by increasing the coherence of the path interval width.

A more quantitative heuristic motivation of Eq. (3.11) is as follows. An infinite number of parallel path segments $i = -\infty, \ldots, -1, 0, 1, \ldots, \infty$ on a workpiece plane is considered, separated by a path interval ω, and a point $\mathbf{p}$ on the workpiece plane in the region of these path segments (cf. Fig. 3.13).

When moving a spray gun along the middle one ($i = 0$) of those segments, a deposit is generated. Slicing the deposit with a plane perpendicular to the segment through $\mathbf{p}$ delivers a profile curve which is represented by a height function h over the line of intersection between the plane with the workpiece plane.

When moving the spray gun along one of the neighboring segments in the same way, the resulting profile function is the same, up to a translation by ω. The deposit $h_\mathbf{p}$ generated at $\mathbf{p}$ by moving the gun along all given path segments is the sum of the values of all profiles at $\mathbf{p}$. The values are of the form $h(x_\mathbf{p} - i \cdot \omega)$, $i = -\infty, \ldots, \infty$, where $x_\mathbf{p}$ is the horizontal

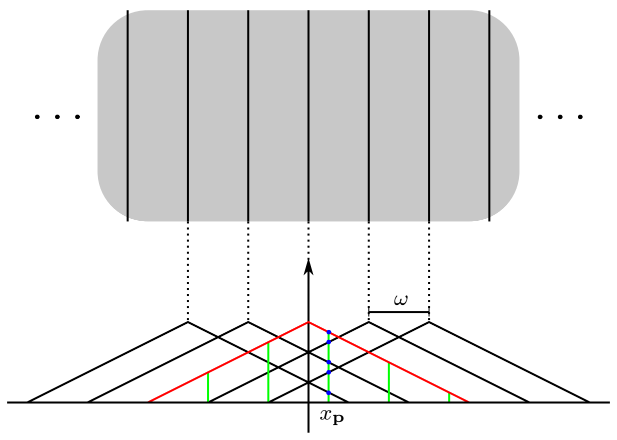


Fig. 3.13: Deposition along parallel path segments on a workpiece plane (top). In the profile (bottom), the coating height at $x_\mathbf{p}$ is the sum of heights of the profiles of all path segments at $x_\mathbf{p}$. The profile of the deposit of a single segment is represented by a triangular shape. The different heights at $x_\mathbf{p}$ can be equivalently represented by the values $h(x_\mathbf{p} - i \cdot \omega)$ at sample points $x_\mathbf{p} - i \cdot \omega$ of the profile h in the center.

coordinate of **p** in a common frame of the profile functions. Thus,

$$h_\mathbf{p} = \sum_{i=-\infty}^{\infty} h(x_\mathbf{p} - i \cdot \omega) \, .$$

Let H be the integral of the profile function h over the interval $[-\infty, \infty]$. Then

$$h_\mathbf{p} = \frac{\sum_{i=-\infty}^{\infty} h(x_\mathbf{p} - i \cdot \omega) \cdot \omega}{\omega} \approx \frac{H}{\omega} \, .$$

This results from considering the sum in the numerator of the second term as a Riemannian approximation of the integral. A rearrangement yields

$$h_\mathbf{p} \cdot \omega \approx H \, .$$

Let $h^s_\mathbf{p}$ be a desired coating height. Then the required path interval satisfies

$$h^s_\mathbf{p} \cdot \omega^s \approx H \, .$$

Both equations together lead to

$$h_\mathbf{p} \cdot \omega \approx h^s_\mathbf{p} \cdot \omega^s \, ,$$

or

$$\omega^s \approx \omega \frac{h_\mathbf{p}}{h^s_\mathbf{p}} \, .$$

This coincides with Eq. (3.11) up to the smoothing operator.

This consideration shows that it is likely that the new path is already a good solution, at least if the paths are not considerably curved. Otherwise, the method can be iterated until a satisfying solution is achieved (step 6). In every iteration, Eq. (3.11) is applied to the result of the preceding step.

The parameters of the initial path may be defined as follows. For the velocity $v_\mathbf{v}$ the approach outlined in the last paragraph of Section 3.6.1 may be employed. The isoline increments at the vertices may all get the same value. The value may be related to the spray gun distance from the surface and the opening angle of its spray cone. The configuration should be so that the spray cone covers several neighboring path segments. The direction of the spray cone may be chosen along the surface normal, or as close to it as possible if collisions prevent this direction.

3.6.3 Evaluation

The evaluation first has been performed on the saddle surface. The desired coating height has the shape of a Gaussian distribution. In the first experiment a non-optimized spray path is used. The seed set of the distance function used for the impact path is the lower boundary of the mesh. The spray gun is moved perpendicularly along this path with constant distance and speed. The spray process is executed with a simulator [111] employing the deposition model of Duncan et al. [35]. The opening angle of the spray cone is 18.33°. This corresponds to a footprint radius on the surface of 33 mm at a gun distance of 100 mm, which is the distance used in the simulation. The resulting coating error is visualized in Fig. 3.14. Due to the non-uniform desired coating height and the uniform spray path the error is considerable.

The gray boundary region, which has approximately the width of the radius of the spray cone on the surface, has been excluded from the calculations and the evaluations. The reason is the method of parametrization of the spray gun path over an impact path which does not allow the gun to move beyond the surface boundary in the required way. This, however, is necessary to achieve a reasonable result close to the boundary. For the application to a given workpiece surface this can be taken into account by extending the surface by a sufficiently wide boundary region. Data required in the boundary region are extrapolated from the relevant region.

The results of the first two iterations of path interval adaptation based on the coating error (Section 3.6.2) with a uniform spray path as initialization are shown in Fig. 3.15(a) and Fig. 3.15(b). For comparison, the result of the approach of path construction from the result of the pathless tool pose optimization by path interval adaptation (Section 3.6.1) is shown in Fig. 3.15(c). Both methods produce good results. The coating-error-based optimization after two iterations is even better than the pathless parameter optimization. Although one iteration is worse than the pathless optimization, it is still acceptable.

The runtime is dominated by the runtime of the distance function calculation, which is 220 ms on an Intel Core i7-2600 processor at 3.4 GHz for the saddle surface mesh made of 22440 triangles. The runtime of the pathless optimization, which is not required for coating-error-controlled path interval adaptation, is 8 min.

Furthermore, the evaluation has been performed the same way on the mechanical engineering part. The seed curve of the distance function is again at the lower boundary of the mesh. The coating error for the initial spray path is visualized in Fig. 3.16(a).

The results of the path interval adaptation based on the coating error for the first, third and fifth iteration are depicted in Fig. 3.16(b) and Fig. 3.17.

The results of the coating-error-controlled path interval adaptation with smoothing are shown in Fig. 3.18. They are somewhat worse than without smoothing, but the generated paths are slightly smoother.

The results based on pathless parameter optimization are shown in Fig. 3.19.

The runtime for the distance function calculation on the mesh with 6170 triangles is 60 ms, while the pathless optimization of the spray time density took 2 minutes.

The methods of this chapter do not make explicit use of the shape of the spray cone. Simulations with different shapes including also asymmetric ones, conducted for coating-

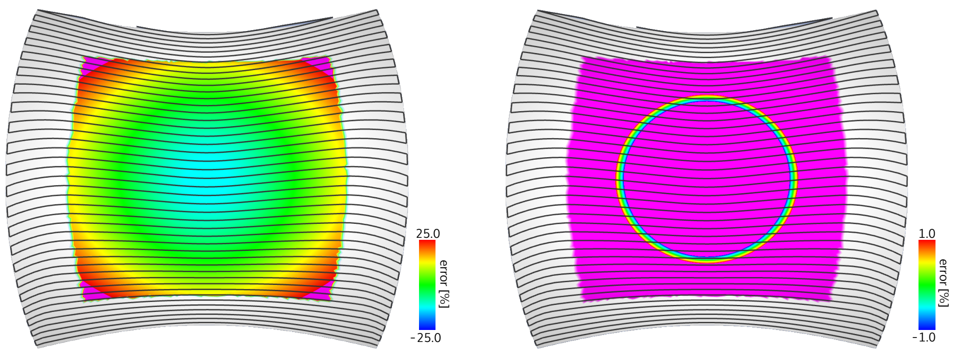


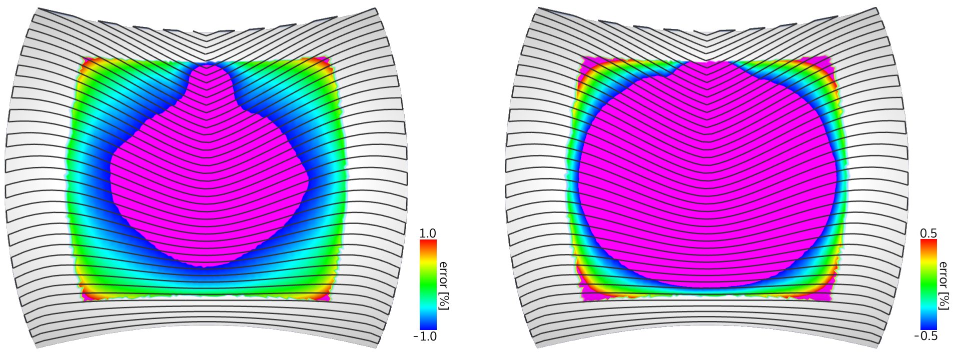
Fig. 3.14: Relative coating error for a Gaussian desired coating height distribution on a saddle surface (cf. Fig. 3.10) resulting from a gun movement at constant speed along a path with a homogeneous path interval. The image on the right shows the same data as the image on the left, but restricted to a smaller error range. The regions with an error outside the range are displayed in magenta. The gray boundary region has been excluded from error considerations. Its width is approximately the radius of the spray cone on the surface.

error-controlled path interval adaptation, have shown the same behavior. This indicates a robustness of the method which might make it useful in a real setting in which the simulation in step 6 of the MTF method is replaced with the physical process (Chapter 3.4). In this case the shape of the spray cone is not known as well.

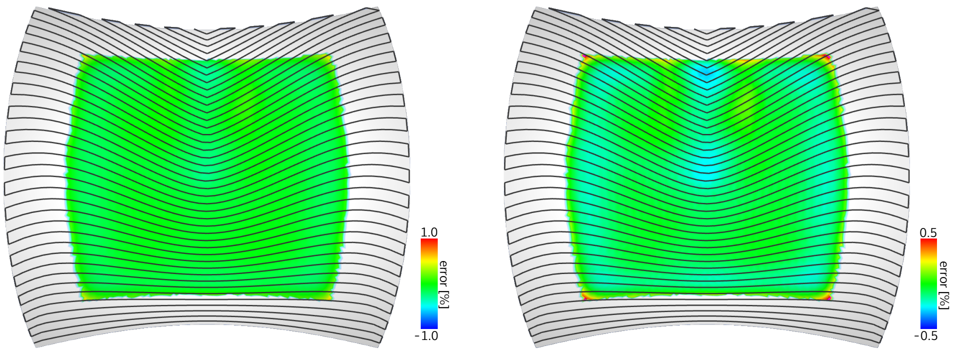
3.7 Concluding Remarks

The flexibility of the approach to represent impact paths by isolines of an appropriate distance function, implemented on workpieces represented by appropriate triangular meshes, has been demonstrated. One central task of the method is the choice of the metric tensor field. In the case studies from milling and spraying presented in this chapter the metric tensor fields have been chosen using models of the processes: the Lin-Koren bound for milling leading to Eq. (3.4), the relation between velocity, path interval size, and spray time density in Eq. (3.9), and the relation between the path interval and coating height used in Eq. (3.11). Future work may extend this to further applications.

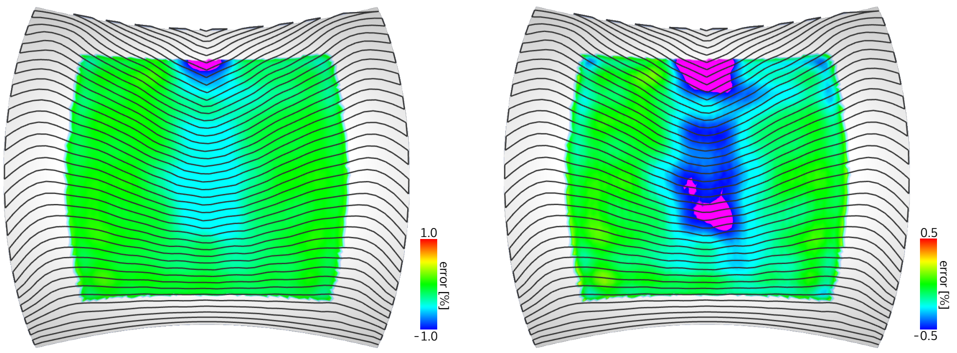
One example is taking into account contact forces for machining processes. Most path planning algorithms for machining neglect dynamic aspects of the machining system [67]. If high, cutting forces may cause tool deflections which e.g. influence the quality of the result. One possibility to cope with this problem is to keep the cutting forces in a feasible range and to avoid considerable variation along the path. An idea which might achieve this goal is to determine the absolute cutting forces along a given distance-function-based impact path by simulation [3]. If the resulting forces exceed a given upper bound, the distance between neighboring path segments has to be decreased, and if they fall below a lower bound, the distance may be increased. The distance adaptation can be achieved by adapting the metric tensors. A possibility to determine the amount of adaptation may be an iterative search loop based on this observation, by alternating adaptation and simulation-based evaluation. A function quantitatively relating the cutting force and path interval size would be helpful in order to keep the number of iterations of this straightforward approach low. The development of such a function is an issue of future research for which existing models of cutting force behavior [68] may serve as a starting point. The aim should be to define the



(a) First iteration. Max: 1.73%, avg: 0.82%, rmse: 0.93%.

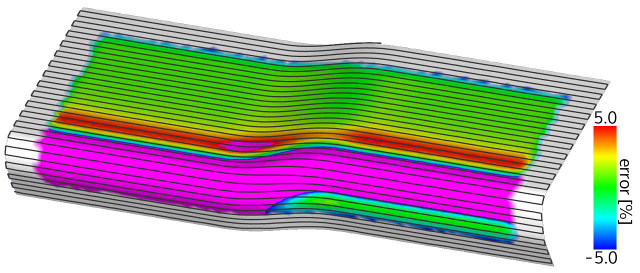


(b) Second iteration. Max: 0.66%, avg: 0.096%, rmse: 0.11%.

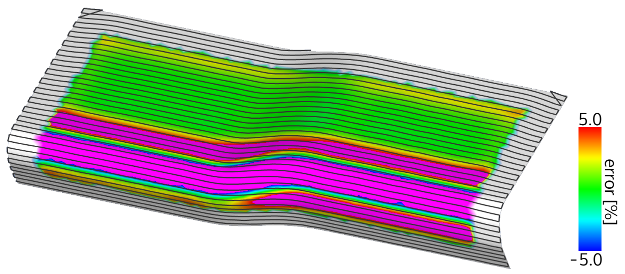


(c) Adaptation to pathless parameter optimization. Max: 1.3%, avg: 0.18%, rmse: 0.25%.

Fig. 3.15: Error of (a) the first and (b) the second iteration of coating-error-controlled path adaptation without smoothing, starting with the path of Fig. 3.14. The error values required for the metric tensors in the gray boundary region have been extrapolated from those of the relevant region. For comparison, the error of spray path calculation for pathless optimization by path interval adaptation is depicted in (c). The right images show the same data as the images left on the left, but restricted to a smaller error range, like in Fig. 3.14.

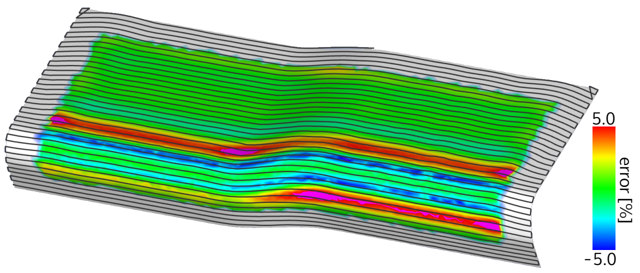


(a) Max: 23.9%, avg: 5.45%, rmse: 8.89%.

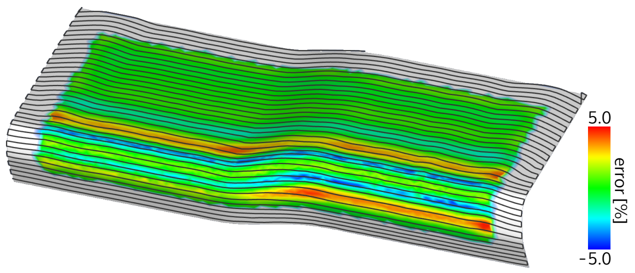


(b) Max: 9.2%, avg: 2.74%, rmse: 3.87%.

Fig. 3.16: Coating height error on a mechanical engineering part for (a) an isotropic path and (b) the first iteration of the coating-error-controlled path interval adaptation without smoothing, analogously to Fig. 3.14.

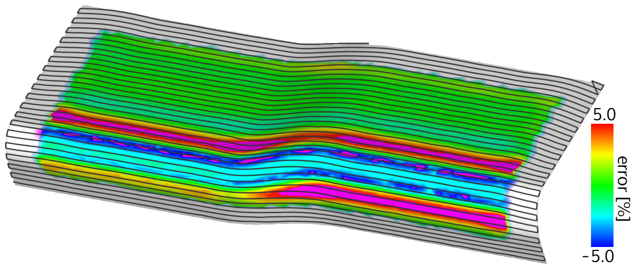


(a) Max: 6.1%, avg: 1.44%, rmse: 2.09%.

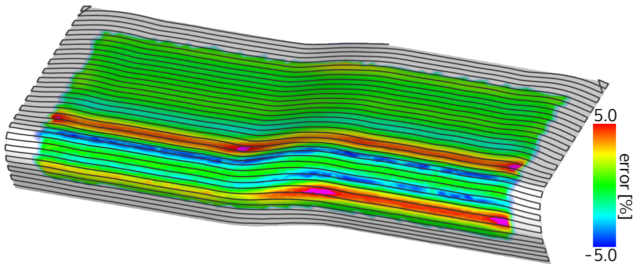


(b) Max: 4.7%, avg: 1.13%, rmse: 1.57%.

Fig. 3.17: Coating height error for (a) the third and (b) the fifth iteration of the coating-error-controlled path interval adaptation without smoothing.

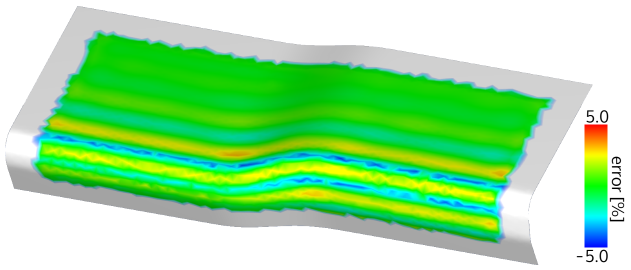


(a) Max: 7.2%, avg: 1.82%, rmse: 2.56%.

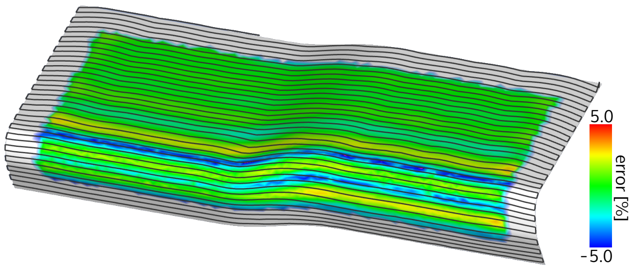


(b) Max: 5.7%, avg: 1.38%, rmse: 1.94%.

Fig. 3.18: Coating height error for (a) the third and (b) the fifth iteration of the coating-error-controlled path interval adaptation with smoothing by convolution. It can be noticed that the error is higher than without smoothing, while the resulting paths are more smooth compared to Fig. 3.17.



(a) Max: 4.4%, avg: 0.955%, rmse: 1.32%.



(b) Max: 5.2%, avg: 1.09%, rmse: 1.56%.

Fig. 3.19: Coating result of (a) the pathless optimization and (b) its transfer to a path by path interval adaptation.

function for arbitrary path directions at a surface point. In contrast to the simulation-based approach, such a function could lead to a path-independent tensor field like the one for cusp-height adaptation.

Another approach to tackle the tool deflection problem is compensation. Compensation may also be achieved by deforming a given path [11]. However, an immediate application of the MTF method seems not to be possible in this case. Possibly an incremental version by successive path-segment-wise adaptation of the distance function might lead to a solution, but this requires considerable further research.

As an alternative to the model-based approach, appropriate tensors might also be found by formulating and solving global optimization problems, like e.g. minimization of the cusp heights for milling and minimization of the coating error for spray coating, over the space of the metric tensor parameters. However, solving such optimization problems can be expected to be computational expensive. As the iteration of Eq. (3.11) for the case study of spray coating has shown, optimization may also be necessary if a model is employed, and thus is included by step 6 of the MTF method. However, optimization may be accelerated by a model, as demonstrated by the low number of iterations sufficient to achieve a reasonable solution.

Another issue of future research is the choice of the seed set in step 1 of the MTF method. Properties of a favorable seed set might also be specified as an optimization problem. Alternatively, process models can be used as well, for instance the approaches to defining process-optimized path directions for cusps by Chiou and Lee [25] and for cutting forces by López de Lacalle et al. [75].

4 Quantitative Improvement of Tool Impact Paths Defined by Isolines of Scalar Functions on Triangular Mesh Workpiece Surfaces

Isolines of distance functions suffer from two properties which have a negative influence on the usefulness of the resulting curves: locations with discontinuous derivatives and local extrema. Those properties may induce sharp corners in isolines and varying distances, and a non-uniformly nested topological structure of isolines, respectively. In Section 3.4.4, an intuitive, qualitative approach of distance function smoothing has been employed to reduce the problem of discontinuous derivatives. In this chapter, optimization-based, quantitative approaches for contour-parallel and direction-parallel offset curves, respectively, are presented to reduce these difficulties. For the contour-parallel case the curvature, the mutual distance, and the topology of the isolines are optimized over a finite-dimensional family of scalar functions derived from the distance function of the contour. In the direction-parallel case objectives including the number, the normal and the geodesic curvature of isolines are optimized over the distance functions of a finite-dimensional family of seed curves. Algorithms to solve these optimization problems on triangular meshes are proposed and employed to demonstrate the usefulness of the methods.

4.1 Introduction

Locations with discontinuous derivatives in distance functions induce sharp corners and varying distances in isolines (Fig. 4.1). This is an issue in particular for contour-parallel curves since these singular locations are induced by curve segments of high curvature. Since in contrast to the direction-parallel case the seed curve is the contour, or at least a curve similar to the contour, such segments are more difficult to avoid for contour-parallel curves. Sharp corners of production paths based on those schemes enforce low tool velocities, and varying distances imply lower distances between neighboring paths to achieve the desired covering. Both have negative effects on the processing time or even on the feasibility of a path.

Another issue is the topology of distance function isolines. In particular for non-convex regions the distance functions may have several peaks which induce a non-uniformly nested structure of isolines (Fig. 3.2(a)). This may cause tool retractions and thus increase the processing time.

This chapter presents two novel methods for the construction of isolines for the contour-parallel and for the direction-parallel case, respectively, which are concerned with those difficulties. Their main idea is to specify the isoline quality quantitatively and solve the resulting optimization problem.

The first method is a *medial-axis-controlled approach to optimizing contour-parallel isolines*. The method is based on a representation of the contour's distance function by a weighted

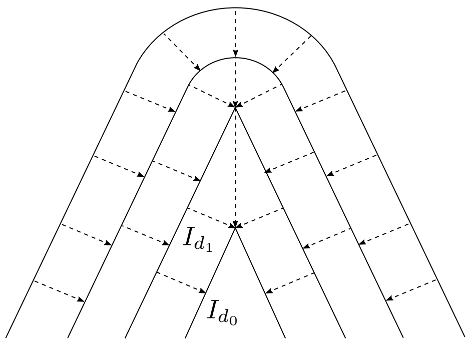


Fig. 4.1: Quasi-parallel isolines. The two inner curves I_{d_1}, I_{d_2} for isovalues d_1 and d_2 have sharp corners and a non-uniform distance. The arrows are directed from a curve point to its nearest neighbor on a neighboring curve.

distance function of the contour's medial axis. This representation opens the possibility to modify the distance function by varying the weights so that an objective function which includes curvature, the mutual distance, and number of peaks of the isolines of the modified distance function is optimized. The related loss of conformity with the contour is remedied by a suitable interpolation method.

The second method is a *medial-axis-controlled approach to optimizing direction-parallel isolines.* The isolines result from a distance function controlled by just a pair of surface points. Its seed set is the medial axis of the two points. The pair of points is chosen to optimize objectives including the normal and the geodesic curvature of isolines, and the number of isolines. The latter aims to minimize the number of isolines required to cover the surface in order to minimize the number of tool turns at the contour of the surface.

Both optimization problems are of multi-objective type. Approaches of reduction to a single-criterion problem as well as a multi-objective Pareto optimization are considered for their solution. For the solution of the contour-parallel case the paradigm of evolutionary optimization is employed. The direction-parallel case is optimized by grid search, which is feasible due to the low dimension of the search space.

The two approaches are implemented for the representation of the workpiece surface by a triangular mesh. In order to calculate weighted distance functions the algorithm by Mitchell et al. [80] used in Chapter 3 is modified appropriately. The usefulness of the approaches and their implementations is demonstrated experimentally.

The following Section 4.2 gives a survey of related work. Section 4.3 is devoted to the concept of weighted distance functions, the definition and calculation of medial axes from distance functions, and the approximation of medial axes of more complex source sets by those of source sets of a finite number of points, with emphasis on triangular meshes. Section 4.4 describes the medial-axes-controlled contour-parallel isoline optimization, while Section 4.5 treats the direction-parallel isoline optimization. Section 4.6 concludes the chapter.

4.2 Related Work

Several attempts have been undertaken to tackle the problems of non-uniformly nested isolines and of sharp corners on isolines. A possibility to avoid points with discontinuous derivatives on isolines is the level-set method [92]. The level-set method constructs a sequence of offset curves iteratively by displacing the current front curve along the gradient of its implicit representation, i.e. its distance function. Non-differentiable sharp corners in curves can be avoided by locally adapting the velocity of displacement of the front curve to its curvature [33, 114]. A drawback of the incremental construction of front curves is a missing global view which would help to reduce the number of non-uniformly nested isolines (cf. Fig. 3.2), or to avoid them at all.

An approach to avoid non-uniformly nested isolines has been presented by Held and Spiegelberger [47]. They describe a method which employs the medial axis of a distance function as guidance for the construction of spiral curves. These curves are not immediately related to the isolines but are at least indirectly based on the concept of distance functions. The approach of contour-parallel isoline construction of this chapter uses medial axes as well, but in a more general way. It can achieve quasi-parallel curves of similar structure as a special case, even on curved surfaces. If spiral curves are desired, they have to be converted.

A further possibility is to replace the distance function with a different kind of scalar function. The approach by Bieterman and Sandstrom [12] for 2D-pocket machining uses a partial differential equation which delivers a scalar function with just one inner extremum, and thus concentric isolines.

The approach by Bouard et al. [18] for 2D-pocket machining focuses on curve smoothing of a given curve pattern whose topology, however, remains unchanged. It specifies the desired properties by an optimization problem with constraints. The method presented in this chapter also employs optimization, but includes the optimization of the curve topology.

A third possibility to avoid the problems is to smooth the distance function by a local smoothing operator in the environment of every vertex, cf. Section 3.4.4. The aim is to reduce the number of local extrema which are responsible for non-uniformly nested isolines. A more flexible possibility is topological smoothing, which is adaptive to the structure of the scalar function. The structure of a scalar function is characterized by local minima, local maxima and saddle points whose topology can be expressed by contour trees, Reeb graphs, or Morse-Smale complexes [99]. Tierny et al. [99] have presented an algorithm which replaces a given scalar function on a triangular mesh with a new one from which a subset of such critical points specified by the user are removed. The resulting scalar function keeps the rest of the critical points and has a minimal deviation from the original one. This way, the topological complexity of isoline nestings may be reduced. The method by Weinkauf et al. [108] additionally delivers a smooth, i.e. differentiable solution. It uses the Morse-Smale complex and requires more efforts of implementation. Smoothing is achieved by optimizing a Laplacian objective function.

The methods presented in this chapter are also based on topological considerations of scalar functions on triangular meshes. However, they take into account constraints concerning geometric properties and the quality of isolines as well. In contrast to the other topological smoothing approaches, which are mostly constructive, they specify the desired properties as a continuous multi-objective optimization problem. Instead of general scalar functions, they focus on distance functions. The required topological information is provided by the medial axis of the seed curve.

4.3 Distance Functions and Medial Axes on Triangular Meshes

This section starts with the extension of distance functions to weighted distance functions. As in other chapters, triangular meshes are chosen as surface representation. This is followed by a presentation of the concept of medial axes. Finally, the approximation of a medial axis of a more complex source set by the medial axis emerging from a source set of a finite number of points is treated.

4.3.1 Weighted Distance Functions

As already defined in Chapter 3, given a set $S \subset M$, called *source* or *seed set*, on a metric surface $\mathcal{M} = (M, d)$, the *distance function* d_S is a function on the surface which gives the distance of any surface point $\mathbf{p}$ to S w.r.t. d. If $\mathbf{p}$ is in S, then the distance is 0. Otherwise, the distance of $\mathbf{p}$ is the infimum of all distances of $\mathbf{p}$ to any point of S,

$$d_S(\mathbf{p}) \coloneqq \inf_{\mathbf{s} \in S} d(\mathbf{p}, \mathbf{s})\,.$$

By modifying the source set, different kinds of distance functions can be obtained. One example employed in the following is that of *additively weighted source sets* inducing *additively weighted distance functions*. In this case the source S is augmented by a function $\sigma(.)$, which assigns an additive real weight $\sigma(\mathbf{s})$ to every element $\mathbf{s} \in S$. A weighted source is denoted by S_σ. The *weighted distance* of a surface point $\mathbf{p} \in M$ to S_σ is defined as

$$d_{S,\sigma}(\mathbf{p}) \coloneqq \inf_{\mathbf{s} \in S_\sigma} \left(d(\mathbf{p}, \mathbf{s}) - \sigma(\mathbf{s})\right).$$

Note that this is a signed distance which may become negative for $\sigma > 0$.

As described in Section 3.3.2, distance functions on triangular mesh surfaces and source sets S of a finite number of surface points can be calculated by the *MMP algorithm*. The algorithm reduces distance and path calculations to those concepts in the plane by unrolling sequences of triangles (Fig. 3.6). Based on this property, it is obvious that the MMP algorithm may be extended to additively weighted source sets by employing the concept of additively weighted Voronoi diagrams in the plane [69, 95]. In this case, the weight of the source point at which a shortest path starts is subtracted from the path length.

4.3.2 Medial Axes

Going back to Blum [13], the medial axis of a two-dimensional plane region can be defined as the set of center points of disks inside the region which touch the region contour at least twice. Alternatively, a medial axis might be characterized as the set of those points in the region for which shortest paths of equal length to at least two different points on the contour exist. A further possibility is to denote a medial axis as the set of those points in the region for which at least two different shortest paths of equal length to the contour exist. All those definitions may be used to extend the definition to two-dimensional regions on curved metric surfaces. However, the results may be different. The result of the first definition depends on the definition of the concept of "disks", and the results of the second and third version may also be different on curved surfaces. In this thesis the following definition is used. Given a metric surface $\mathcal{M} = (M, d)$ and a source set $S \subset M$, the *medial axis* $\mathcal{A}_S$ is the set of points $m \in M$ for which shortest paths of equal length from m to at least two different points on S exist. This definition covers the case of the medial axis of a bounded region by taking the contour of the region as S.

Apart from potentially existing degenerate cases, the *medial axis in a bounded region* on a metric surface is a graph consisting of a finite number of nodes which are connected by non-intersecting curve segments as edges (Fig. 4.2(a)). The curve segments are induced by surface points with shortest paths of equal length to exactly two points of the source set.

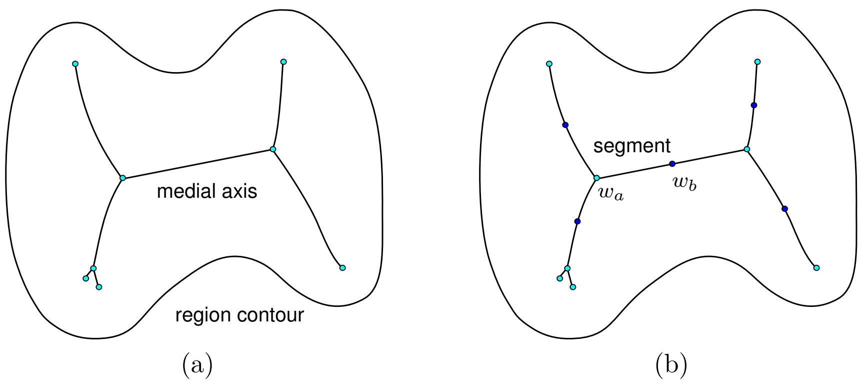


Fig. 4.2: (a) Qualitative structure of medial axes of bounded regions. (b) Approximation of the weights of a weighted medial axis by a weight function controllable by a finite set of node weights w_i. The weights on a segment are linearly interpolated between the weights of its end points.

They either end at nodes of degree greater than two which are points with more than two such shortest paths, or are open-ended in which case the limit point is added as a vertex of degree 1. Vertices of degree 1 are denoted as *leaves*, and those of degree greater than 2 as *branching vertices*.

4.3.3 Approximate Calculation of Medial Axes

The medial axis of a finite set S of points as a source set, according to the definition of the previous section, and the Voronoi diagram of S (cf. Section 3.3.2) are identical. Recall that the Voronoi diagram consists of all those points whose distance to at least two source points is equal. Since the distance of two points is defined as the length of a shortest path between two points, a point on the boundary of a Voronoi diagram has at least two different shortest paths of equal length to S.

For a finite source set S of points on a mesh surface, the Voronoi diagram can be calculated from the data structure delivered by the MMP algorithm. In the original paper of Mitchell et al. [80] the distance function is interpreted as an (implicit) Voronoi diagram, because it can answer the question which the nearest source is. Liu et al. [73] have explicitly presented an algorithm which derives a Voronoi diagram from a geodesic distance function on a triangular mesh by using the MMP algorithm. After preprocessing the mesh the algorithm identifies triangles containing Voronoi vertices, and traces the Voronoi edges from there. However, it ignores that Voronoi vertices can have more than three incident Voronoi edges and that Voronoi vertices can lie on mesh vertices or edges. It also ignores that Voronoi edges can go through mesh vertices, which invalidates their triangle classification. For that reason an alternative approach is proposed in Appendix C which remedies those limitations.

The algorithm to calculate a Voronoi diagram for a point set opens the possibility to approximate the medial axis of more complex source sets. A Voronoi diagram of a finite set of sample points on the contour of a region as source set may have curve segments completely inside, completely outside, or intersecting the region contour. The interesting part consists of the curve segments in the interior of the region. Figure 4.3(a) shows a magnified view of a cut-out of the Voronoi diagram of a fine point sampling of the contour of Fig. 3.7(c) and (d). The sampling consists of 1140 points of equal distance. Figure 4.3(b) shows a subset of the

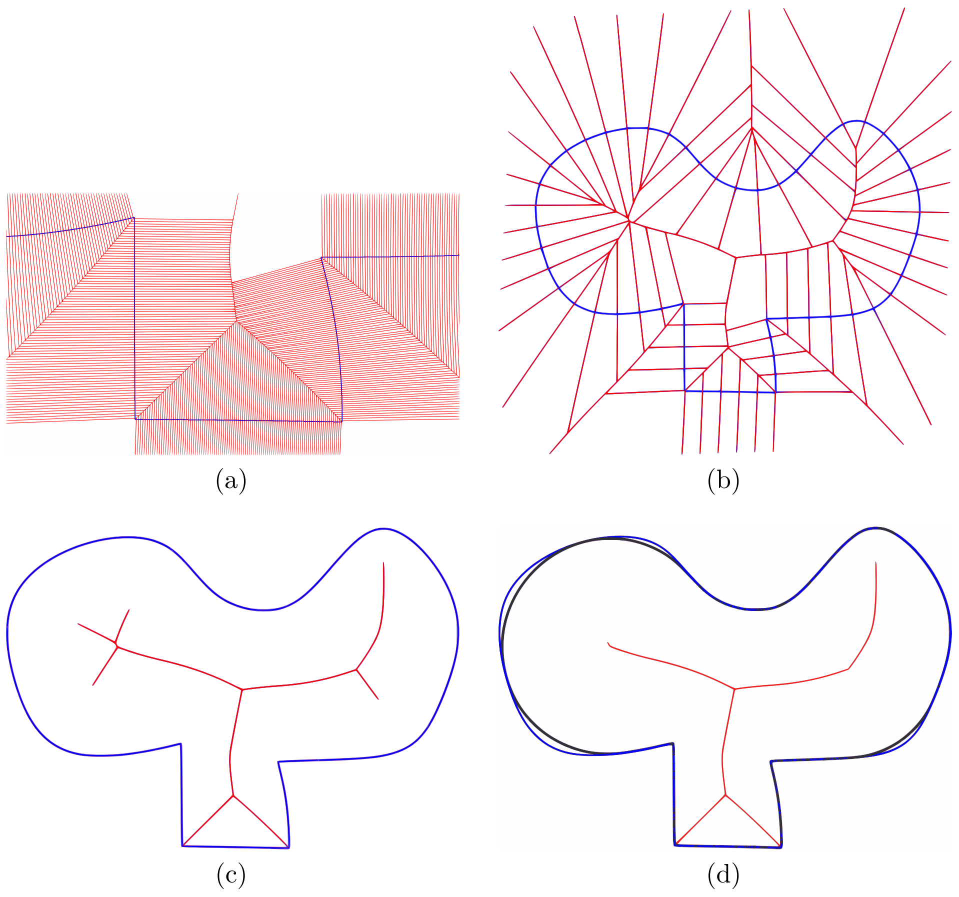


Fig. 4.3: A bounded region with a medial axis induced by a dense sequence of sample points on the region contour. (a) A magnified view of a cut-out of the Voronoi diagram. (b) A Voronoi diagram subset by removing a significant number of those edges from the Voronoi diagram which are induced by neighboring sample points on the contour. (c) The approximated medial axis generated by removing all axis segments intersecting the region contour. (d) Pruning of the medial axis of (c). The black curve shows the zero-isoline of the distance function of the distance-weighted medial axis, cf. Section 4.4.2.

Voronoi diagram by removing a significant number of those edges from the Voronoi diagram which are induced by neighboring sample points on the contour. Figure 4.3(c) displays the approximated medial axis by removing all Voronoi edges which intersect with the contour of the region or are outside the region. As can be noticed in the figure, this is a reasonable way to approximate the medial axis of a region by a subset of the Voronoi diagram of a point set representation of the contour. The Voronoi edges intersecting the region contour are Voronoi edges between two neighboring points and therefore do not guarantee that two different shortest paths to the contour exist. The approximation is of course very dependent on the point sampling density and the curvature of the contour between the samples.

In Section 4.4.2 and 4.5.1, medial axes will be used to control the shape of distance functions. One operation will be pruning of medial axes.

4.4 Medial-axis-controlled Contour-parallel Isoline Optimization

Medial-axis-controlled contour-parallel isoline optimization is based on the observation that the distance function induced by the region contour can also be generated as additively weighted distance function of the medial axis provided with suitable weights (cf. Section 4.4.1). By using those weights as control parameters, and possibly additional pruning of the medial axis (cf. Section 4.4.2), the shape of the weighted distance function and thus of the resulting isolines are optimized according to several objectives (cf. Section 4.4.4). The objectives include the minimization of the number of isoline peaks, a reasonable path interval variation, and smooth paths. The latter two objectives are expressed by using the gradient length and the absolute value of the Laplace operator. Additionally, a smooth behavior of the weights along the medial axis is pursued.

A drawback of pruning and changing weights for optimization is that the isolines of the resulting weighted distance functions need no longer be contour-parallel, i.e. the zero-isolines may be different from the region contour (Fig. 4.3(d)). Thus, the constraint of contour parallelism is taken into account in an additional step (Fig. 4.5, cf. Section 4.4.3).

The objectives are optimized either by combining them to a single objective as a weighted sum or directly by Pareto optimization. In both cases optimization is performed by an evolutionary algorithm (cf. Section 4.4.5). The behavior of the algorithms is experimentally analyzed (cf. Section 4.4.6).

4.4.1 Equivalent Representation of Distance Functions of Boundaries

The optimization builds upon the medial axis of the distance function d_S of the region contour as source set S. A weighted distance function $d_{\mathcal{A}_S}$ of the medial axis $\mathcal{A}_S$ of S is defined by using d_S on $\mathcal{A}_S$ as additive weight,

$$d_{\mathcal{A}_S,d}(\mathbf{p}) := \inf_{\mathbf{m}\in\mathcal{A}_S} (d(\mathbf{p},\mathbf{m}) - d(\mathbf{m},\mathbf{s}(\mathbf{m}))) \tag{4.1}$$

where $\mathbf{p}$ is an arbitrary surface point and $\mathbf{s}(\mathbf{m})$ a point of the source set S to which $\mathbf{m}$ has minimum distance (Fig. 4.4(a)).

Under the hypothesis that for every surface point $\mathbf{p}$ a shortest path from a medial axis point $\mathbf{m}(\mathbf{p})$ to S exists which contains $\mathbf{p}$, the weighted distance function $d_{\mathcal{A}_S,d}$ coincides with the original distance function d_S up to its sign, i.e.

$$d_{\mathcal{A}_S,d}(\mathbf{p}) = -d_S(\mathbf{p})\,, \tag{4.2}$$

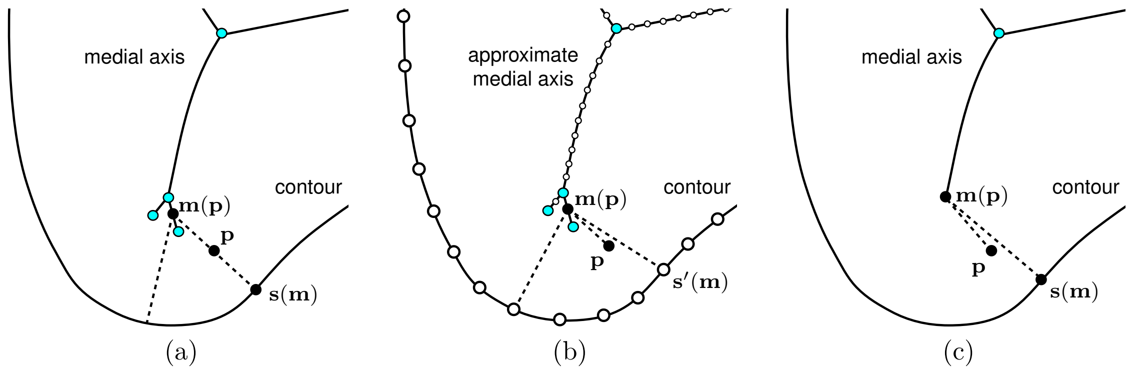


Fig. 4.4: Illustrations of properties of (a) $d_{\mathcal{A}_S,d}$, (b) $d_{\mathcal{A}_S,d}$ with the approximated medial axis, and (c) $d_{\widehat{\mathcal{A}}_S,d}$.

for the following reason. By the metric triangle equation and metric symmetry ($d(\mathbf{p},\mathbf{m}) = d(\mathbf{m},\mathbf{p})$),

$$d(\mathbf{m},\mathbf{s}(\mathbf{m})) - d(\mathbf{p},\mathbf{m}) \leq d(\mathbf{p},\mathbf{s}(\mathbf{m}))\,.$$

By the hypothesis,

$$\begin{aligned} d_{\mathcal{A}_S,d}(\mathbf{p}) &= -(d(\mathbf{m}(\mathbf{p}),\mathbf{s}(\mathbf{p})) - d(\mathbf{p},\mathbf{m}(\mathbf{p}))) \\ &= -d(\mathbf{p},\mathbf{s}(\mathbf{p})) = -d_S(\mathbf{p}) \end{aligned}$$

holds, where $\mathbf{s}(\mathbf{p}) = \mathbf{s}(\mathbf{m}(\mathbf{p}))$ is the end point of the path in S. The right equation is immediate. To show the left equation, let $\hat{\mathbf{m}}$ be any other medial axis point. Then

$$\begin{aligned} d(\hat{\mathbf{m}},\mathbf{s}(\mathbf{p})) - d(\mathbf{p},\hat{\mathbf{m}}) &\leq d(\hat{\mathbf{m}},\mathbf{p}) + d(\mathbf{p},\mathbf{s}(\mathbf{p})) - d(\mathbf{p},\hat{\mathbf{m}}) \\ &= d(\mathbf{p},\mathbf{s}(\mathbf{p})) = d_S(\mathbf{p})\,. \end{aligned}$$

Hence the supremum over the left side, which is $-d_{\mathcal{A}_S,d}(\mathbf{p})$, is equal to $d_S(\mathbf{p})$.

A heuristic argument for the correctness of the hypothesis in a bounded region R and its contour curve as source set S is as follows. Let $\mathbf{s}(\mathbf{p}) \in S$ be the end point of a shortest path from a point $\mathbf{p} \in R$ to S, $\mathbf{p}$ not on the medial axis. The path at $\mathbf{p}$ is geodesically extended in a locally shortest way until a point $\mathbf{m}(\mathbf{p})$ is reached for which a path of different direction at $\mathbf{m}(\mathbf{p})$ to S exists whose length is equal to the distance of $\mathbf{m}(\mathbf{p})$ and $\mathbf{s}(\mathbf{p})$. This means that $\mathbf{m}(\mathbf{p})$ is on the medial axis, and $\mathbf{p}$ is on a shortest path from $\mathbf{m}(\mathbf{p})$ to S. A point $\mathbf{m}(\mathbf{p})$ of this property exists since the distances of all points in the bounded region R to S are bounded by a constant.

The distance function $d_{\mathcal{A}_S,d}$ can be approximately calculated as follows. First, the original distance function of a finite set S' of sample points on the contour S, as well as the approximation of the medial axis $\mathcal{A}_S$ based on S' as described in Section 4.3.3 (cf. Fig. 4.4b) are determined. The approximate medial axis is represented by a sufficiently dense finite set M' of sampling points. Then the distance function of M' is obtained by the MMP algorithm according to the original surface metric. This distance function allows to calculate the terms $d(\mathbf{p},\mathbf{m})$ of (4.1), while the terms $d(\mathbf{m},\mathbf{s}(\mathbf{m}))$ are available from the medial axis or the distance function of the contour, respectively.

4.4.2 Control Parameters

The optimization of contour-parallel isolines is controlled by modifying the weighted medial axis $\mathcal{A}_{S,d}$ defined in the previous section. Two mechanisms are introduced in the following, *medial axis pruning* and *weight variation*.

Medial axis pruning is a method to shape the medial axis by removing leaves, i.e. vertices of degree 1, including their incident edge. Leaf removal can be iterated by further removing leaves emerging in a pruned graph. If the medial axis is a tree, this way the medial axis may be reduced down to just one vertex. If the medial axis contains cycles, the minimum is a graph without leaves which is homotopic to the region. Since pruning reduces the size of a medial axis, it may contribute to improve computational efficiency. The weighted distance function of a pruned medial axis $\widehat{\mathcal{A}}$ derived from $\mathcal{A}$ is

$$d_{\widehat{\mathcal{A}}_{S,d}}(\mathbf{p}) := \inf_{\mathbf{m}\in\widehat{\mathcal{A}}_S} (d(\mathbf{p},\mathbf{m}) - d(\mathbf{m},\mathbf{s}(\mathbf{m}))) \tag{4.3}$$

where $\mathbf{p}$ is an arbitrary surface point and $\mathbf{s}(\mathbf{m})$ is a point of the source set S to which $\mathbf{m}$ has minimum distance (Fig. 4.4(c)). On triangular meshes, it can be calculated from a point sampling of the pruned approximate medial axis by the MMP algorithm.

The zero-isoline of $d_{\widehat{\mathcal{A}}_{S,d}}$, in the following denoted by $\hat{S}$, will in general not be equal to S. If S is the contour of a region R, $\hat{S}$ is the contour of the region $\hat{R}$ in which the isolines of $d_{\widehat{\mathcal{A}}_{S,d}}$ are contour-parallel. $\hat{R}$ is a subset of the region R bounded by S.

In practice, pruning should be performed in a way which keeps the difference between R and $\hat{R}$ small. A criterion of removal can be based on the quotient of the difference of the maximum and minimum distance value on the medial axis edge, divided by the edge length. If the quotient is greater than a given threshold, then the edge is removed.

A variant of medial axis pruning is *medial branch pruning*. Branch pruning is used to shorten the curve segment incident to a leaf by taking a point somewhere on the curve segment as new leaf. It is useful at sharp corners of boundaries where a complete leaf pruning may eliminate too much of the original shape.

Weight variation, the second mechanism of shape control, modifies the weight function $d(\mathbf{m},\mathbf{s}(\mathbf{m}))$ of $\mathcal{A}_{S,d}$, or of a pruned version $\widehat{\mathcal{A}}_{S,d}$ of it, into a new weight function $w(\mathbf{m})$. For example, local maxima of the additive weights on the medial axis are responsible for peaks. Thus, eliminating a maximum by adapting it to its environment may eliminate a peak. An extreme case is that all additive weights are equal to the maximum weight. Then all peaks are eliminated, and the isolines are contour-parallel to the medial axis (Fig. 4.5(a)).

The weighted distance function of a pruned medial axis $\widehat{\mathcal{A}}_S$ is defined as

$$d_{\widehat{\mathcal{A}}_{S,w}}(\mathbf{p}) := \inf_{\mathbf{m}\in\widehat{\mathcal{A}}_S} (d(\mathbf{p},\mathbf{m}) - w(\mathbf{m})) \tag{4.4}$$

where $\mathbf{p}$ is an arbitrary surface point, $\mathbf{s}(\mathbf{m})$ is a point of the source set S to which $\mathbf{m}$ has minimum distance, and $w(\mathbf{m})$ is the given weight of $\mathbf{m}$. On triangle meshes it can be calculated by the MMP algorithm like $d_{\widehat{\mathcal{A}}_{S,d}}$.

For the optimization, weight variation is reduced to *weight functions controllable by a finite set of parameters*. The set of parameters consists of the weights w_i of a finite number of control points $\mathbf{m}_i$ on the pruned medial axis (Fig. 4.2(b)). The set of control points includes the nodes of $\widehat{\mathcal{A}}_S$, and further points on the medial axis. The interpolating function is composed of linear functions defined over the curve segments of the pruned medial axis between incident control points. The linear functions on the segments linearly interpolate the weights w_a, w_b at the segment endpoints $\mathbf{m}_a$, $\mathbf{m}_b$.

A useful initial solution of iterative optimization approaches is an *approximation by a finitely controllable weight function* of $\widehat{\mathcal{A}}_{S,d}$. Such an approximation is derived by firstly adding the nodes of $\widehat{\mathcal{A}}_{S,d}$ to the set of control points, with their given weights. Then points with maximum deviation from the distances interpolated between two neighboring control points are inserted in order to iteratively split segments into two parts until the desired quality of approximation is reached. The inserted points are added to the set of control points. This way, control points are inserted just where required. Experience shows that, by combination of pruning and this procedure, the number of parameters can be kept low so that the approach is practically feasible. A reason is that small changes of distances at the medial axis due to interpolation usually induce just small changes of the distance function.

4.4.3 Constraint of Contour Parallelism

The control parameters of Section 4.4.2, *medial axis pruning* and *weight variation* do not take the constraint of contour parallelism into account (cf. Fig. 4.5(a)). In this section, an interpolation method is presented to remedy this problem. The input is

- the original distance function $d_{\widehat{\mathcal{A}}_{S,d}}$ of a non-pruned or pruned medial axis,
- an optimizing weighted distance function $d_{\widehat{\mathcal{A}}_{S,w}}$, derived from $d_{\widehat{\mathcal{A}}_S}$ by weight variation according to Section 4.4.2.

The pruned medial axis $\widehat{\mathcal{A}}_S$ has to have a positive distance to the source set S, i.e. $\inf_{\mathbf{s}\in S,\ \mathbf{m}\in\mathcal{A}_S} d(\mathbf{m},\mathbf{s}) > 0$, in order to avoid troubles with zero-nominators.

The output is a distance function $\hat{d}_{\widehat{\mathcal{A}}_{S,w}}$ which is a modification of $d_{\widehat{\mathcal{A}}_{S,w}}$ satisfying the constraint of contour parallelism.

The desired distance function with contour-parallel isolines is obtained by defining a parameter of linear interpolation between $d_{\widehat{\mathcal{A}}_{S,w}}$ and 0 according to

$$\hat{d}_{\widehat{\mathcal{A}}_{S,w};\hat{S}}(\mathbf{p}) := \alpha_{\hat{S}}(\mathbf{p}) \cdot d_{\widehat{\mathcal{A}}_{S,w}}(\mathbf{m}(\mathbf{p}))\,, \tag{4.5}$$

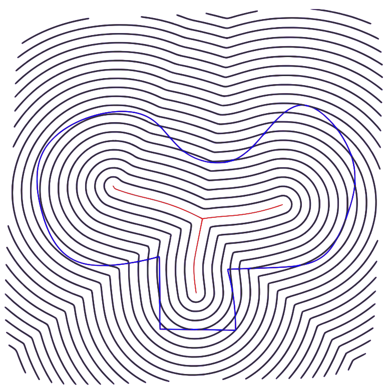
with

$$\alpha_{\hat{S}}(\mathbf{p}) := \frac{d_{\widehat{\mathcal{A}}_{S,d}}(\mathbf{p})}{d_{\widehat{\mathcal{A}}_{S,d}}(\mathbf{m}(\mathbf{p}))}\,, \tag{4.6}$$

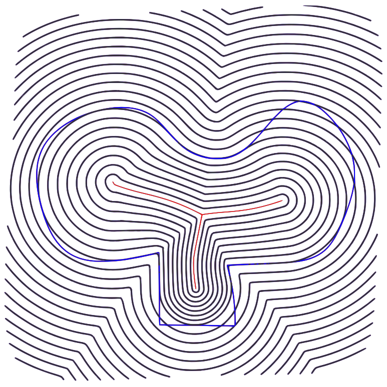
$$\mathbf{m}(\mathbf{p}) = \arg \inf_{\mathbf{m}\in\widehat{\mathcal{A}}_S} \left(d(\mathbf{p},\mathbf{m}) - d(\mathbf{m},\mathbf{s}(\mathbf{m}))\right), \tag{4.7}$$

where $\mathbf{s}(\mathbf{m})$ is a point of the source set S to which $\mathbf{m}$ has minimum distance, cf. (4.3). For $\widehat{\mathcal{A}}_S = \mathcal{A}_S$, $\mathbf{m}(\mathbf{p})$ is the start point of a shortest path from the medial axis $\mathcal{A}_S$ to S through $\mathbf{p}$, cf. Section 4.4.1. The value $d_{\widehat{\mathcal{A}}_{S,w}}(\mathbf{m}(\mathbf{p}))$ at one endpoint of the path is interpolated with the value 0, which implies contour parallelism, at the other endpoint of the path. $\mathbf{m}(\mathbf{p})$ can be obtained as a point closest to $\mathbf{p}$ on the pruned medial axis $\widehat{\mathcal{A}}_S$ according to the distance function $d_{\widehat{\mathcal{A}}_{S,d}}$, i.e. the point $\mathbf{m}$ corresponding to the value of $d_{\widehat{\mathcal{A}}_{S,d}}(\mathbf{p})$, cf. (4.3) and Fig. 4.4.

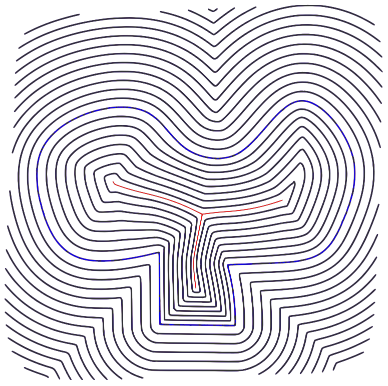
For a pruned medial axis $\widehat{\mathcal{A}}_S$, $\alpha_{\hat{S}}(\mathbf{p}) = 0$ holds for $\mathbf{p}$ on the zero-isoline $\hat{S}$ of $d_{\widehat{\mathcal{A}}_{S,d}}(\mathbf{p})$. Hence the isolines of $d_{\widehat{\mathcal{A}}_{S,w}}$ are contour-parallel for the region $\hat{R}$ bounded by $\hat{S}$. For $\mathbf{p} \in \widehat{\mathcal{A}}_S$, $\mathbf{m}(\mathbf{p}) = \mathbf{p}$, which can be shown analogously to the proof of (4.2). Hence $\hat{d}_{\widehat{\mathcal{A}}_{S,w}}(\mathbf{p}) = d_{\widehat{\mathcal{A}}_{S,w}}(\mathbf{p})$ holds for $\mathbf{p} \in \widehat{\mathcal{A}}_S$.



(a)



(b)



(c)

Fig. 4.5: Contour parallel isolines derived from the distance function of a weighted pruned medial axis $\widehat{\mathcal{A}}_{S,w}$ with a constant weight equal to the minimum value of $d_{\widehat{\mathcal{A}}_{S},d}$ on $\widehat{\mathcal{A}}_S$. (a) The isolines of the original distance function of the pruned medial axis. (b) Contour parallelism by interpolation with respect to the contour $\hat{S}$ induced by $\widehat{\mathcal{A}}_{S,d}$. (c) Contour parallelism by interpolation with respect to the original contour S.

Using

$$\alpha_S(\mathbf{p}) := \frac{d_{\mathcal{A}_S,d}(\mathbf{p})}{d_{\widehat{\mathcal{A}}_S,d}(\mathbf{m}(\mathbf{p}))} \tag{4.8}$$

instead of $\alpha_{\hat{S}}(\mathbf{p})$ in (4.5) leads to a distance function $\hat{d}_{\widehat{\mathcal{A}}_S,w;S}$ with isolines in parallel to the original contour S. Furthermore, $d_{\mathcal{A}_S,d}(\mathbf{p}) = d_{\widehat{\mathcal{A}}_S,d}(\mathbf{p})$ for $\mathbf{p} \in \widehat{\mathcal{A}}_S$, so that $\hat{d}_{\widehat{\mathcal{A}}_S,w}(\mathbf{p}) = d_{\widehat{\mathcal{A}}_S,w}(\mathbf{p})$ holds as well.

Figure 4.5(a) shows the isolines of the distance function of a pruned medial axis $\widehat{\mathcal{A}}_{S,w}$ with a constant weight equal to the minimum value of $d_{\widehat{\mathcal{A}}_S,d}$ on $\widehat{\mathcal{A}}_S$. Figure 4.5(b) and (c) present the results of the two variants of contour parallelism by interpolation.

As already outlined before, on triangular meshes the required distance functions of the weighted medial axes can be calculated by the modified MMP algorithm. The medial axes are represented by a finite, sufficiently dense set of sampling points annotated by their weights. $\mathbf{m}(\mathbf{p})$ required in (4.6) and (4.8) is available at the source point of the window which the MMP algorithm uses to calculate $d_{\widehat{\mathcal{A}}_S,d}(\mathbf{p})$.

4.4.4 Objective Functions

The objectives of optimization concern the smoothness of the distance function, the gradient length in the distance function and the number of peaks. The optimization is performed over the distance functions of a weighted pruned medial axes according to Section 4.4.2 which are made contour-parallel with the interpolation method of Section 4.4.3.

The *smoothness of the distance function* is evaluated by the geodesic curvature $\bar{\kappa}_g(\mathbf{v}_i)$ of the isolines at the mesh vertices $\mathbf{v}_i$ of the region R of interest. The objective function to be minimized is

$$f_{\kappa,g} = \frac{1}{\sum_{\mathbf{v}_i \in R} A_i} \sum_{\mathbf{v}_i \in R} A_i \cdot (\bar{\kappa}_g(\mathbf{v}_i))^2 ,$$

where A_i is the area of the Voronoi region of vertex $\mathbf{v}_i$ inside the surface region R. The definition and a possibility of calculating $\bar{\kappa}_g(\mathbf{v}_i)$ can be found in Appendix B.

The *occurrence of peaks* is related to the minima of the additive weights along the medial axis, located between peaks. If there are no minima, then generally no peaks occur on the medial axis and likewise in the whole region. In order to reduce minima on the medial axis the objective function demands that for all pairs of neighboring nodes of a node i on the medial axis, the interpolation of their additive weights at i should be smaller than the actual weight w_i at i, and that leaf weights should also be greater than the interpolation between their neighbor and the contour. This is mathematically expressed by

$$\begin{aligned} f_{\text{top}} = & \sum_i \sum_{j,k \in N_i, j \neq k} \frac{1}{\binom{|N_i|}{2}} \cdot \left(\min\left(0, w_i - w_{ijk}^{\text{int}}\right)\right)^2 \\ & + \sum_{i \text{ is leaf}, j \in N_i} \left(\min\left(0, w_i - w_{ij}^{\text{int}}\right)\right)^2 , \end{aligned}$$

with

$$\begin{aligned} w_{ijk}^{\text{int}} &= \frac{w_j \cdot d(\mathbf{m}_i, \mathbf{m}_j) + d(\mathbf{m}_i, \mathbf{m}_k) \cdot w_k}{d(\mathbf{m}_i, \mathbf{m}_j) + d(\mathbf{m}_i, \mathbf{m}_k)} , \\ w_{ij}^{\text{int}} &= \frac{w_j \cdot d(\mathbf{m}_i, \mathbf{m}_j)}{d(\mathbf{m}_i, \mathbf{m}_j) + |d_{\widehat{\mathcal{A}}_S, d}(\mathbf{m}_i)|} , \end{aligned}$$

where $\mathbf{m}_i$ are nodes on the medial axis, and $d(\mathbf{m}_i, \mathbf{m}_j)$ is the distance along the medial axis between nodes i and j. The first term represents the case of inner nodes, and the second term the case of leafs. Note that the denominator of the leaf case shows just one term since the second one is equal to 0 because it corresponds to a point on the contour.

The *distance of two neighboring isolines* might be measured by the Hausdorff distance. The Hausdorff distance might be approximately calculated using two point sets generated by point sampling the two isolines. However, the calculation of the distances between the two point sets on a surface, e.g. by the MMP algorithm, is time consuming and relatively sophisticated.

An efficient alternative is to consider the gradient field of the scalar function generating the isolines. The length of gradient vector at any point on the mesh characterizes the local distance between neighboring isolines. If the length of the gradient is the same everywhere, two neighboring isolines have equal distance everywhere. This is e.g. the case for smooth distance functions (i.e. in C^1 everywhere) where the gradient length is 1. A higher gradient means that the curves come closer to each other, while a lower gradient indicates a higher distance. Based on this observation, objectives concerning the distance of neighboring curves may be formulated as objectives on the gradient length.

Algorithm 3 Medial-axis-controlled contour-parallel isoline optimization

```
Input: A surface with boundary, represented by a triangular mesh.
Output: A set of contour-parallel isolines.
1: Define a pruned medial axis.
2: Define an initial discretized weight function on the pruned medial axis.
3: Set up the objective function (single objective version or multi-objective version).
4: Choose the weighting of the sub-objective functions (single-objective version).
5: Optimize the objective function by an evolutionary algorithm.
6: Select a solution from the Pareto front (multi-objective version).
7: Determine the set of isolines from the solution.
```

The gradient $\nabla d_{\widehat{\mathcal{A}}_S,w}$ of the distance function $d_{\widehat{\mathcal{A}}_S,w}$ on a triangular mesh can be calculated as described in Appendix A.

The objective of a uniform isoline interval length is expressed by the distance of the gradient length to the interval $[1,2]$, summed up for all vertices inside the surface region R and not part of a triangle touching the medial axis:

$$f_{\text{grad}} = \frac{1}{\sum A_i} \sum_{\mathbf{v}_i \in R} A_i \cdot \max(0, \|\nabla d_{\widehat{\mathcal{A}}_S,w}(\mathbf{v}_i)\| - 2, \\ 1 - \|\nabla d_{\widehat{\mathcal{A}}_S,w}(\mathbf{v}_i)\|)^2 \,.$$

This means that a gradient length between 1 and 2 is evaluated with 0, and that a gradient length outside this interval is penalized with increasing distance to the interval.

A further objective is *uniform gradient length along the medial axis*, analogously to the approach of Held et al. [47]. It is desirable because the medial axis is in general the place where the gradient length is smallest if it exists, and a uniform gradient along the medial increases the minimal gradient length and distributes the isoline intervals evenly on the surface. This includes the gradient length from the leaves to the contour. The goal of a uniform gradient along the medial axis is expressed in the objective function

$$f_{\text{iso}} = \sum_i \sum_{j,k \in N_i, j \neq k} \left(\frac{|w_j - w_i|}{d(\mathbf{m}_j, \mathbf{m}_i)} - \frac{|w_k - w_i|}{d(\mathbf{m}_k, \mathbf{m}_i)} \right)^2 \\ + \sum_{i \text{ is leaf}, j \in N_i}^{n} \left(\frac{|w_j - w_i|}{d(\mathbf{m}_j, \mathbf{m}_i)} - \frac{|w_i|}{d(\mathbf{m}_i, \mathbf{s}(\mathbf{m}_i))} \right)^2 ,$$

where $\mathbf{m}_i$ are nodes on the medial axis with weights w_i, and $\mathbf{s}(\mathbf{m}_i)$ is a point on the zero-line of $d_{\widehat{\mathcal{A}}_S,d}$ closest to $\mathbf{m}_i$.

4.4.5 Setting up and Solving the Optimization Problem

The steps of calculating a family of contour-parallel isolines by medial-axis-controlled optimization are compiled in Algorithm 3.

In this work, a suitably pruned medial axis is chosen in advance (step 1), and the optimization is restricted to the weight values as degrees of freedom. Pruning is performed manually for the weighted medial axis calculated as described in Section 4.4.1. Initial weighting (step 2) is performed by transferring the weights of the initial non-pruned medial axis, or by interactive weight assignment.

The objectives of the preceding section lead to a multi-objective optimization problem which may be treated in two ways (step 3). One approach is to define a single objective function as a weighted sum of all objectives,

$$f = c_{\kappa,g} \cdot f_{\kappa,g} + c_{\text{top}} \cdot f_{\text{top}} + c_{\text{grad}} \cdot f_{\text{grad}} + c_{\text{iso}} \cdot f_{\text{iso}} \,, \tag{4.9}$$

with weights $c_{\cdot}$. Preferences may be expressed by higher weights. The weights are chosen in advance (step 4).

Alternatively, the *Pareto set* (*Pareto front*) of optimal feasible solutions may be determined. The *Pareto set* contains all solutions with the property that for the objective values $(f_{\kappa,g}, f_{\text{top}}, f_{\text{grad}}, f_{\text{iso}})$ no further solution exists with at least one entry less than, and all entries less or equal to those in this vector.

Both versions of the optimization problem are solved by evolutionary algorithms (step 5). The basic idea of *evolutionary algorithms* is to consider individuals, each of which represents a feasible solution. A fitness value is assigned to every individual by evaluation of a fitness function for the individual. From a first generation of finitely many individuals an evolutionary algorithm generates a sequence of further generations by applying evolutionary operators with the aim to increase the fitness of the individuals. Typical evolutionary operators are selection, mutation, and recombination which, applied to the individuals of a generation, generate the next generation.

For the optimization problem considered here an individual represents a feasible solution by its values of the control parameters of optimization (cf. Section 4.4.2), i.e. the weight values of the pruned medial axis. The fitness function is the objective function of the optimization problem which can be evaluated using the information represented by an individual.

This basic principle is often modified to improve the solutions and the convergence speed of evolutionary algorithms for specific applications. The combined objective function (4.9) is optimized with the evolutionary algorithm CMA-ES [45]. It uses a self-adapting mutation operator which modifies the covariance matrix of a multi-variate normal distribution.

For multi-objective optimization, the evolutionary algorithm MO-CMA-ES [51, 53, 97, 106] implemented in the Shark library [52] is selected. The MO-CMA-ES combines the covariance matrix adaption mutation strategy [45] with non-dominated sorting and the contributing hypervolume as selection strategies to form a multi-objective optimization algorithm. The covariance matrix adaptation is a popular mutation strategy for real-valued function optimization. Non-dominated sorting partitions the population into successive Pareto fronts, while the hypervolume-indicator is used to rank subsets of individuals in the same Pareto front according to the hypervolume they contribute. A desired solution is selected interactively from the resulting Pareto set (step 6).

The isolines are calculated from the resulting scalar function (step 7) according to Section 3.3.3. The desired spacing between isolines is defined by the user by providing appropriate isovalue increments.

4.4.6 Experimental Results

The evaluation presented in the following has used the triangular mesh surface of Fig. 3.7(c) and (d). The mesh consists of 16641 vertices and 32768 faces. Furthermore, the pruned medial axis of Fig. 4.3(d) has been chosen. It has to be emphasized that its asymmetric pruning is of theoretical nature in that it combines several artifacts in order to get insight in the behavior of optimization. On the left side, the medial axis is considerably pruned, in contrast to its other branches. This implies that it loses influence on the distances of the isolines in that region. In the lower part, the medial axis comes very close to the contour because of the two sharp corners.

Optimization is performed with respect to the objective functions of Section 4.4.4, with contour parallelism relative to the original contour S.

The finitely controllable weight function employed has 18 nodes. The weights of the leaf nodes at the two sharp corners are set to 0 and are excluded from optimization. Hence, there remain 16 degrees of freedom.

A population size of 250 has been chosen for evolutionary optimization, and 50000 function evaluations have been executed. The initial population is generated from an initial individual by mutation by a normal distribution with standard deviation of 1. Two variants of initial individuals have been considered. The first one uses a constant weight on the medial axis equal to the maximum distance of the pruned medial axis to the contour. The second one applies a piecewise linear approximation of the distance-weighted pruned medial axis as described in Sect 4.4.2. The runtime of the two optimization processes was approximately 330 minutes for each one on a single core of an Intel Core i7-2600 processor at 3.4 GHz. It has to be noted that the quite high number of individuals of a generation, which is responsible for the high runtime, has been chosen in order to get dense Pareto fronts to understand their structure.

Figure 4.6 visualizes the two Pareto fronts formed by the final generations of individuals in a scatter plot. The Pareto front for the constant weights is shown in red, while the Pareto front for the distances as starting values is drawn in blue. The scatter plot consists of projections of the four-dimensional set on the plane spanned by every pair of coordinates. Hence, the projections usually do not show the Pareto property. Nonetheless, the scatter plots of f_{top} against f_{iso} (view 5) and of $f_{\kappa,g}$ against f_{iso} (view 3) nearly show a rather simple convex Pareto front in two dimensions. The scatter plot of $f_{\kappa,g}$ against f_{top} (view 1) only exhibits this behavior below a value of about 0.007 for $f_{\kappa,g}$, while above that level both objectives show a linear relation. That also explains some similarities between the scatter plots of $f_{\kappa,g}$ against f_{grad} (view 2) and of f_{top} against f_{grad} (view 4).

The rectangular shape of some scatter plots is due to the cropping in some objectives.

Figure 4.7 shows the isolines belonging to the extremal solutions with respect to the different objectives and a solution with balanced objective values. Note that especially the objectives which crop a value by maximization or minimization with zero may have multiple extremal solutions. The extremal solution for $f_{\kappa,g}$ in Fig. 4.7(a) has rather smooth isolines which corresponds to the minimization of the geodesic curvature. The extremal solution for f_{top} in Fig. 4.7(b) only has one minimum, i.e. one peak, while the extremal solution to f_{grad} in Fig. 4.7(c) constrains the gradient length to the interval $[1, 2]$. The extremal solution to f_{iso} in Fig. 4.7(d) shows a rather uniform absolute gradient along the medial axis, without considering the number of minima responsible for peaks, which is caused by taking the absolute values in f_{iso}. The balanced solution in Fig. 4.7(e) is chosen such that it has only one minimum and average values for the other objectives. Compared to the original distance function $d_{\widehat{\mathcal{A}}_S,d}$ in Fig. 4.7(f) the three peaks are removed while the gradient increased.

Two further experiments concern the optimization of the combined objective function (4.9) with contour parallelism relative to the original contour S. The first experiment has been based on the same pruned medial axis as before (Fig. 4.3(d)). The same finitely controllable weight function has been employed, with 16 degrees of freedom. The weights were $c_{\kappa,g} = 100$, $c_{\mathrm{top}} = 30$, $c_{\mathrm{grad}} = 300$, $c_{\mathrm{iso}} = 1$.

The second experiment has involved the pruned medial axis of Fig. 4.5. The finitely controllable weight function had 11 nodes, and hence 11 degrees of freedom.

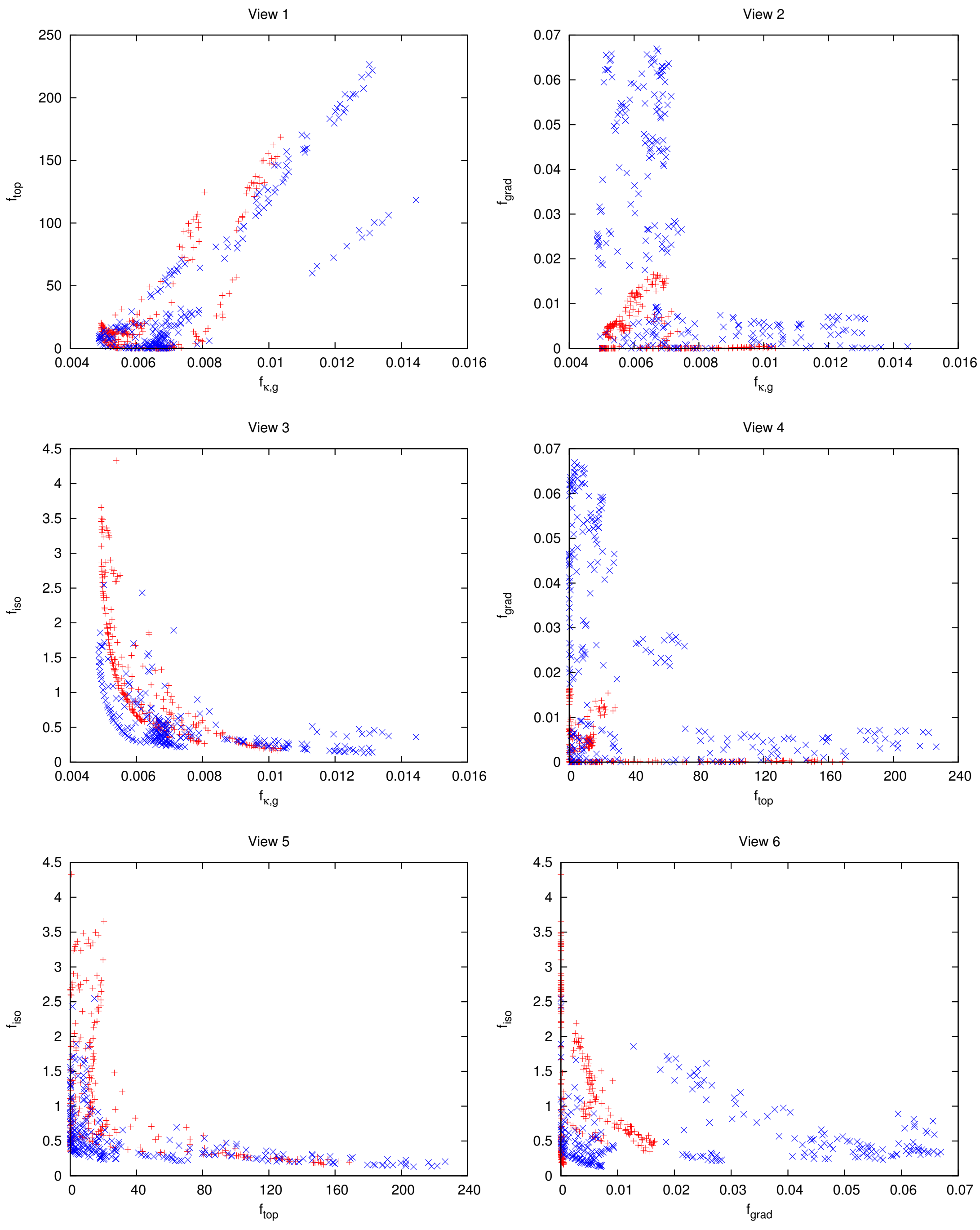


Fig. 4.6: Contour-parallel isoline optimization for the pruned medial axis of Fig. 4.3(d): Scatter-plot visualization of the result of two applications of evolutionary Pareto optimization with different initializations. The two Pareto sets generated by the final individuals are drawn in red and blue.

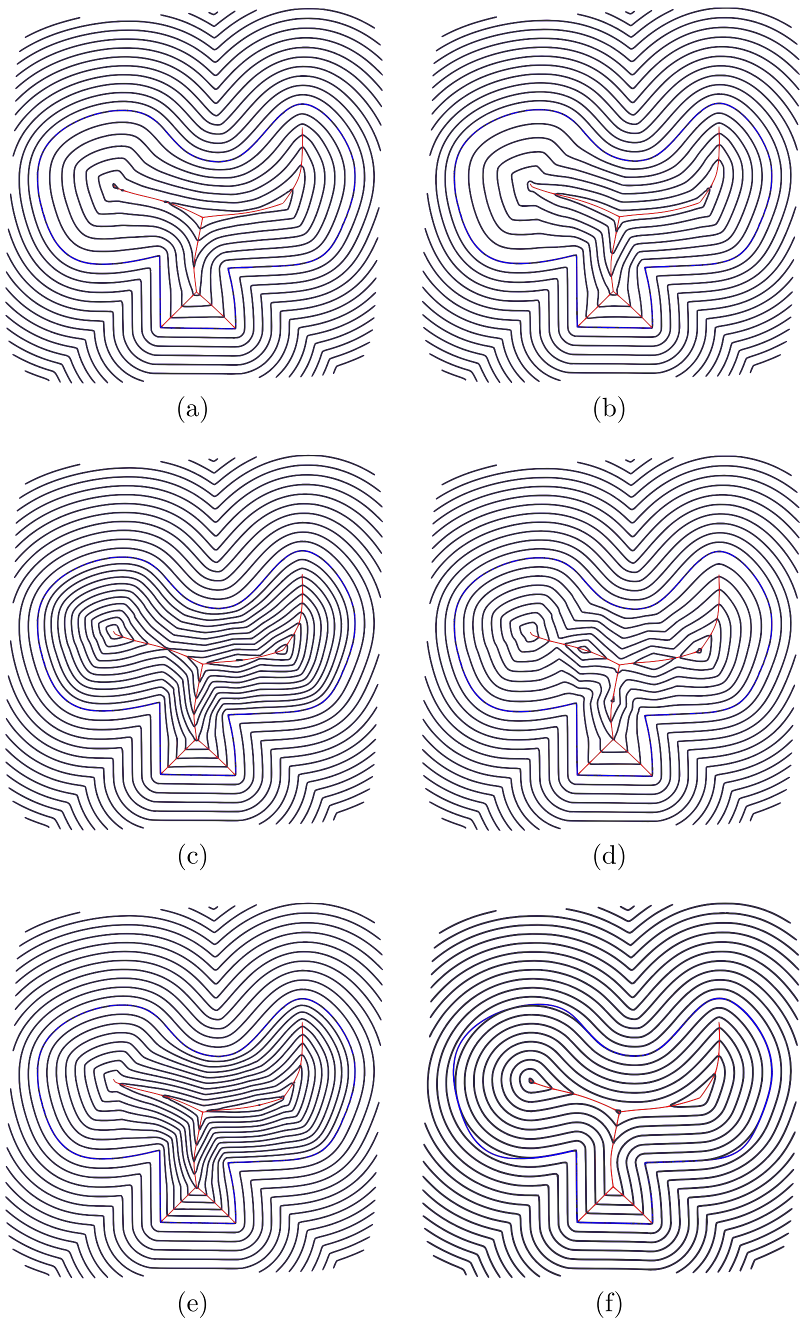


Fig. 4.7: Examples from the Pareto front of Fig. 4.6. (a) Minimized $f_{\kappa,g}$. (b) Minimized f_{top}. (c) Minimized f_{grad}. (d) Minimized f_{iso}. (e) A balanced solution. (f) For comparison: the isolines of $d_{\widehat{\mathcal{A}}_S,d}$ before optimization.

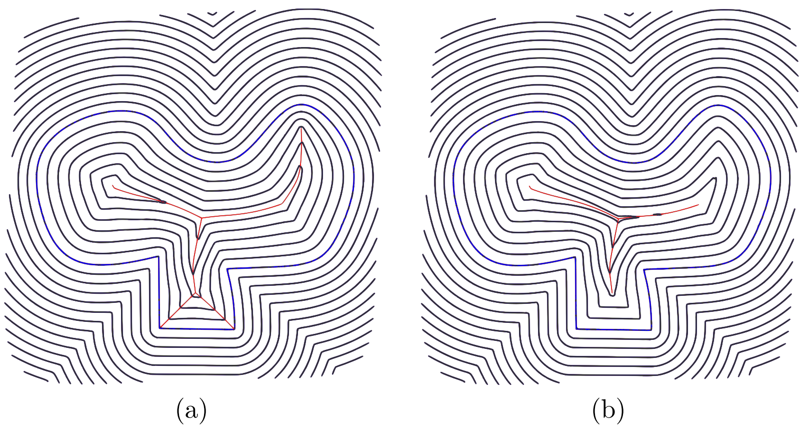


Fig. 4.8: Optimization of the combined objective function with contour parallelism relative to the original contour S, (a) for the pruned medial axis of Fig. 4.3(d), (b) for the pruned medial axis of Fig. 4.5.

Figure 4.8(a) and (b) show the resulting isolines. Both images show that the peaks have been reduced, and that the isolines are quite uniformly distributed.

In these cases the population size has been 12 and 11, and 2000 function evaluations have been executed. The initial standard deviation was chosen as 2. The runtime has been approximately 12 minutes on the same processor as before.

4.5 Direction-parallel Isoline Optimization

The objectives of optimization of direction-parallel isolines are partially different from those of contour-parallel curves. Peaks are no longer relevant since they are mainly caused by the constraint of contour parallelism. Furthermore, the interval variation between neighboring curves can be expected to be non-relevant for surfaces of low variation in the direction-parallel case, so that it is not mandatory to consider this aspect in the objective function.

However, the surface contour is still important. For the direction-parallel case, the number of curves ending at the contour should be minimized. The reason is that a surface leaving path induces a turn of the tool and thus increases the production time.

Furthermore, more flexibility is given with respect to the curvature of paths. For contour-parallel curves, the path curvature is mainly influenced by the surface contour. In contrast to this, the curvature behavior of the surface can be taken into account when choosing the direction for direction-parallel curves. The normal and geodesic curvature of the isolines are used as optimization criteria because in general a low curvature is favorable for the robot or machine tool.

The basic idea of the following approach to optimization of direction-parallel isolines is to choose a class of seed curves depending on a finite number of control parameters (cf. Section 4.5.1). The medial axis of a pair of surface points is chosen for this purpose. Hence, the pair of points are the control parameters of optimization.

The objective functions are defined on the set of the isolines induced by a finite number of isovalues on the distance function of a seed curve (cf. Section 4.5.2). They consider the cardinality of the set of isolines, the normal curvature, and the geodesic curvature of the induced isolines. The objectives are either linearly combined or directly Pareto-

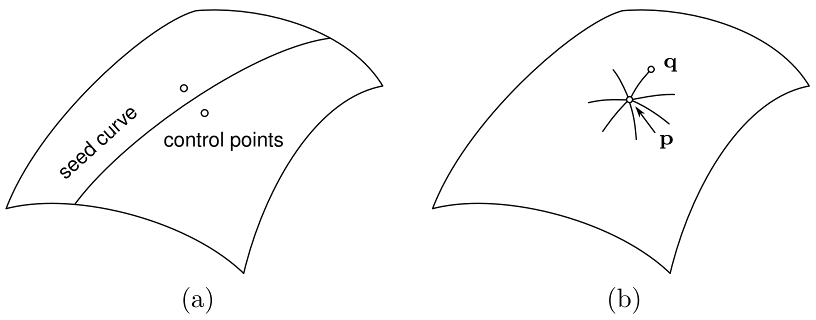


Fig. 4.9: (a) Definition of a seed curve as the medial axis of a pair of surface points as control points. (b) Definition of sample points on a geodesic circles as end points of geodesic curves having the length of the radius and starting at the center of the circle in different directions.

optimized. Optimization is performed by a grid search (cf. Section 4.5.3). The performance is experimentally evaluated (cf. Section 4.5.4). The details are presented in the following.

4.5.1 Parameters

As already mentioned, the seed curve serving as a source of the distance function is constructed as the medial axis of two points on the surface (Fig. 4.9(a)). A main advantage of this approach is the extremely low number of degrees of freedom at a nevertheless high flexibility of the resulting family of seed curves. The medial axis provides an adaptation to the surface and a rather low geodesic curvature.

4.5.2 Objective Function

The objective function of direction-parallel isolines is optimized for a fixed sequence of isovalues d_i, $i = 1, \dots, n(S)$, usually defined by $d_i = i \cdot \Delta d$, with a constant increment $\Delta d > 0$. The number $n(S)$ of curves depends on the seed curve S. Every isoline is represented by a sequence of $m_i(S)$ sample points where $m_i(S)$ depends on S as well.

The quality of a direction-parallel path is described by three objective functions:

- the number of isolines
$$f_{\#}(S) = n(S)\,,$$
- the sum of the absolute normal curvatures at the sample points of the isolines
$$f_{\kappa,n}(S) = \sum_{i=1}^{n(S)} \sum_{j=1}^{m_i(S)} |\kappa_{nd}^{i,j}(S)|\,,$$
- the sum of the absolute geodesic curvatures at the sample points of the isolines
$$f_{\kappa,g}(S) = \sum_{i=1}^{n(S)} \sum_{j=1}^{m_i(S)} |\kappa_{gd}^{i,j}(S)|\,.$$

Here $\kappa_{nd}^{i,j}$ and $\kappa_{gd}^{i,j}$ denote the discrete normal and geodesic curvature of isoline i at position j, cf. Appendix B. For noise reduction, a few consecutive curvature values can be grouped and summed up before taking the absolute value.

```
Algorithm 4 Direction-parallel isoline optimization
Input: A surface with boundary, represented by a triangular mesh.
Output: A set of direction-parallel isolines.
1: Set up the objective function.
2: Choose the weighting of the sub-objective functions (single-objective version).
3: Optimize the objective function (done by the system).
4: Select a solution from the Pareto front (multi-objective version).
5: Determine the set of isolines from the solution.
```

4.5.3 Setting up and Solving the Optimization Problem

The steps of calculation of direction-parallel isolines by optimization are shown in Algorithm 4.

As before, this optimization problem may be treated directly as a multi-objective problem by searching for Pareto-optimized solutions, or by transferring it to a single-objective problem by optimizing the weighted sum

$$F(S) = c_\# \cdot f_\#(S) + c_{\kappa,n} \cdot f_{\kappa,n}(S) + c_{\kappa,g} \cdot f_{\kappa,g}(S) \tag{4.10}$$

of the three objectives with non-negative weights $c_\#$, $c_{\kappa,n}$, and $c_{\kappa,g}$ (steps 1 and 2).

Due to the low dimension of the parameter space, optimization is performed in both cases by grid sampling pairs of sample points as control values of the seed curve S (step 3). The pairs of points over which optimization is performed are obtained as follows. First, a finite number of sample points $\mathbf{p}$ is distributed on the surface e.g. with Turk's remeshing algorithm [102]. Then, for each sample point $\mathbf{p}$, a finite set of points $\mathbf{q}$ is determined by sampling of a geodesic semicircle around $\mathbf{p}$, and the desired pairs of points are obtained as $(\mathbf{p}, \mathbf{q})$.

A geodesic circle with center $\mathbf{p}$ and radius r according to the definition employed is the set of all points $\mathbf{q}$ for which a geodesic path of radius r starting at $\mathbf{p}$ exists. Details for geodesic paths on triangular meshes may be found at Polthier and Schmies [86]. The desired sampling points on the semicircle are obtained as the end points of a finite number of geodesic paths in a finite number of directions covering the angular interval $[0, \pi]$ uniformly (Fig. 4.9(b)). In the direction of favorable solutions, i.e. Pareto optimal solutions, the search is refined by checking additional sample directions.

Selection of a solution from the Pareto set (step 4) and calculation of isolines (step 5) are done like for the contour-parallel case (Section 4.4.5).

4.5.4 Experimental Results

A mechanical engineering part from deep drawing serves as demonstrator for the seed curve optimization, cf. Fig. 4.11. Optimization has been performed in the Pareto way using grid sampling, as described. 33 sample points were randomly spread over the surface. The geodesic semicircle around every sample point was uniformly sampled in 20 directions, and another 20 directions were used around the Pareto-optimal directions.

In Fig. 4.10, the result of evaluation of every pair of points is represented by a 3D point whose coordinates represent the achieved values of the three sub-objective functions. The Pareto-optimal solutions are shown in blue. The spectrum of Pareto optimal solutions ranges from solutions where the number of isolines is the dominant optimization criteria and solutions where one of the curvature criteria is dominant. The resulting distance functions

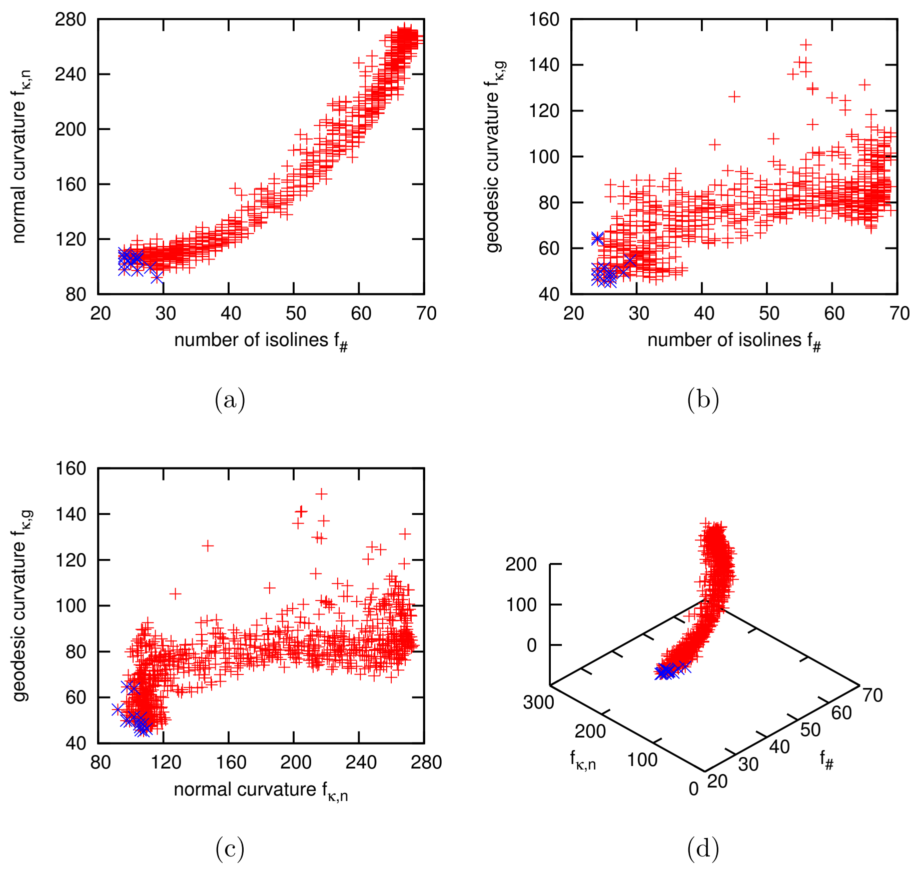


Fig. 4.10: Set of solutions of the direction-parallel isoline optimization for the demonstrator in three different projections (a-c) and in 3D (d). The Pareto optimal solutions are marked in blue.

and isolines of two Pareto optimal solutions are shown in Fig. 4.11. They are considerably different and underline the effect of the different objectives.

The runtime of optimization observed for this example is about 23 minutes. The main cost is the distance function calculation which takes about 500 milliseconds on the mesh made of 24025 triangles.

4.6 Concluding Remarks

An approach to diminish the problems of discontinuous derivatives and local extrema of distance functions based on optimization has been presented. Discontinuous derivatives are responsible for sharp corners in isolines, and local extrema induce a multiple topological nesting of families of isolines. Both effects are unfavorable for production paths. Objectives of optimization include the curvature of isolines, the distance between isolines, the topology of isolines, the normal and geodesic curvature of isolines, and the number of isolines. These criteria are appropriately selected for contour-parallel isolines and direction-parallel isolines, respectively. In both cases, the distance function has been controlled by a finite set of points and associated weights as parameters of optimization, related to suitable chosen medial axes. Reasonable solutions are achieved even for a quite low number of parameters. Methods

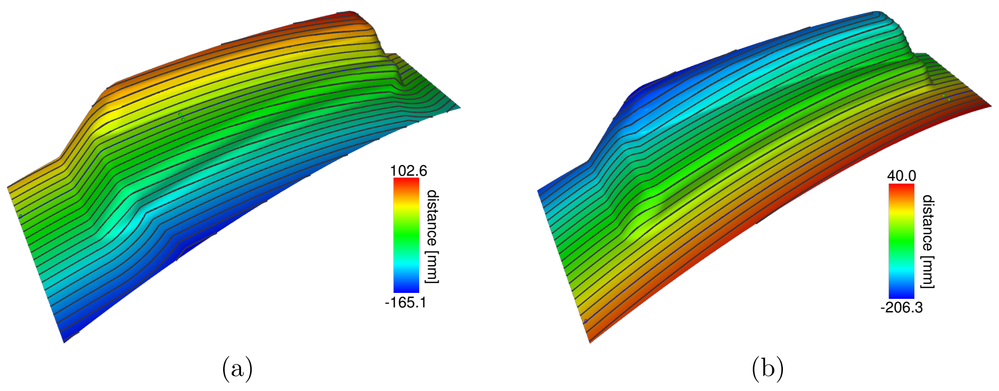


Fig. 4.11: Two Pareto optimal solutions for the demonstrator, one with more weight on the normal curvature (a) and one with more weight on minimizing the number of isolines (b).

and algorithms have been presented for the implementation of the approach on triangular meshes.

A first aspect of future research may be the inclusion of medial axis pruning as a further degree of freedom in the optimization process. The benefit may be a further improved curve quality. Related to this aspect is the question for more specific and efficient approaches to solving the optimization problem, e.g. local optimization techniques for faster convergence. In this chapter, the paradigm of evolutionary algorithms has been used as a tool of flexible prototyping. Nevertheless, the evolutionary approach itself also might be made more efficient. For example, a question is how far the cardinality of the population may be reduced for Pareto optimization in order to still get interesting solutions. A further aspect is interactive optimization using the control points and weights, respectively, for example for further optimization of calculated solutions. An advantage of interactive optimization is that goals may be achieved which cannot be specified with reasonable efforts as formal objective function. Another related aspect is semi-interactive optimization by interactive steering of the optimization process. For example, the objective of optimization of the combined variant of sub-objectives might be changed during the computational optimization process by interactive adaptation of the weights of the sum of sub-objectives during the optimization process.

5 Manipulator Placement Optimization for Path-based Manufacturing Processes

The problem of manipulator placement considered in this chapter is to find a placement of a manipulator relative to a tool path so that it is able to execute the tool path in an optimized way with respect to certain constraints and objectives. Two optimization approaches are employed to solve this problem. The first approach is to separate the placement and path optimization. For manipulators without redundant axes, a wide spectrum of existing optimization methods, including systematic sampling, evolution, and local descent, is analyzed for suitability to the problem on test cases from spray coating. The second approach directly combines manipulator path optimization with the placement. Two novel methods are presented which replace the static position of the workpiece and the strict path-following constraint regarding the tool path, respectively, by an objective (soft constraint) in order to facilitate the optimization. The relaxed problems have the structure of a path optimization problem with a manipulator with redundant axes, which is why this approach is also applicable to redundant manipulation tasks. A solution based on a quasi-Newton method is presented. The evaluation shows that the approaches are practically useful.

5.1 Introduction

Most approaches of path planning for specific production technologies focus on the tool path and neglect the tool guiding manipulator or take it into account by qualitative heuristics. However, for complex workpieces the constraints resulting from the capabilities of the manipulator have to be considered. Unfortunately, the inclusion of the manipulator in the planning process increases the computational complexity of the planning problem because of the additional degrees of freedom, objectives and constraints.

Motivated by this observation, this chapter considers the problem of finding a position of the workpiece relative to the manipulator so that the manipulator can execute a given tool path at all, and if so, in an optimized way. The optimization includes various objectives like the manipulability, which is numerically quantified by measures taken from literature. The main contribution of this chapter is the presentation and the computational analysis of a wide spectrum of algorithmic solution methods of the optimization task emerging from this problem, with the aim to get insight into the computational tractability. Beyond existing methods, two novel methods employing relaxation are introduced.

First, the approach to separate the placement and path optimization is considered. This means that the optimization is primarily performed over the domain of placements, and that for every considered placement in the course of solution the manipulator path, i.e. the motion of the manipulator, is optimized. This approach is particularly natural in the case of manipulators without redundant axes, which is the most common type. In this case, the path optimization problem becomes much more convenient to solve. The methods applied for placement optimization include regular sampling and Latin hypercube sampling (LHS), the evolutionary approaches $(\mu + \lambda)$-ES and CMA-ES, in a single and a multi-objective version, and the local iterative methods of Nelder-Mead and gradient descent (finite differences, response surface method). The best solutions of the experimental evaluation of those methods on test cases from spray coating have been achieved by the evolutionary approaches and the Nelder-Mead algorithm, with the additional advantage of a lower running time of the evolutionary approach due to a better suitability for parallelization. Figures 5.15 and 5.16 show examples of solution.

A particular issue of the separated placement optimization is the constraint to execute the given tool path. This may lead to a very small region of feasible solutions or even none at all, which may result in the algorithm to fail. Moreover, even a complex topology of the region of feasible solutions may impede some of the local optimization methods. An approach to cope with this problem is to relax hard constraints to soft constraints in the form of additional penalty terms in the objective function. Two novel methods of relaxation are proposed. The first one gives up the static location of the workpiece and allows it to move during the execution of the tool path. In this way, the problem of collisions, which is a major reason for infeasibility, is reduced. The original demand of a static location of the workpiece is expressed by a penalty function as part of the objective function. The method is worked out by modeling it as a path optimization problem, which is solved by a quasi-Newton method. The evaluation for the case of spray coating shows that the approach is practically useful.

The second method of relaxation is to replace the strict path-following constraint regarding the tool path with a penalty function punishing the deviation from the tool path. This approach is interesting for spray coating where further control parameters besides the path geometry are available to optimize the deposition. The related optimization problem is basically of the same type as the preceding one so that it can be solved analogously. Another important advantage of the relaxed approach is the potential to find a placement even if there is no feasible solution for the given tool path. This is achieved because the tool path does not have to be followed exactly.

The following Section 5.2 gives a survey on related work. Section 5.3 presents the formal specification of the manipulator placement problem as an optimization problem with constraints. Section 5.4 gives an overview over solution approaches, related definitions, and the design of the experimental analysis. The subsequent sections are devoted to systematic sampling (Section 5.5), evolutionary approaches (Section 5.6), local descent (Section 5.7), and the relaxation methods (Sections 5.8 and 5.9). Section 5.10 discusses the results, and Section 5.11 concludes the chapter.

5.2 Related Work

The aim of *optimal manipulator path planning* or *trajectory planning* is to automatically design a control function for a manipulator, which, when executed, enables the manipulator to optimally solve a task under the constraints caused by its kinematics and dynamics. A classical manipulator task is to move the end-effector from one point to another in space. Objectives include the minimization of the cycle-time and the energy consumption. Fundamental aspects of optimal path planning are path finding with the particular challenge of collision avoidance [26] and manipulator control as special problem of optimal control [9]. Considerable fundamental research has been performed with respect to those topics, which is taken into account in this chapter. The review [5] gives a survey of general solution methods with respect to point-to-point manipulator paths.

An important aspect related to optimal manipulator path planning is *task placement*. The aim of task placement is to develop methods of placing a task to be performed in the workspace of the manipulator so that it can be executed in an optimized way. In this chapter, *path placement* as special version of task planning is considered. The input of a path placement problem is a tool path. This means that the path of the end-effector is completely predefined, in contrast to the general point-to-point task where the planning of the paths between (two) successive points is part of the task placement problem. Path placement is particularly important for manipulators without redundant axes that leave no degrees of freedom for path optimization.

The publications about the placement problem differ with respect to manipulator aspects, i.e. the type of manipulator and the objectives, application aspects, and optimization methods.

A central aspect with respect to manipulation are the objectives of path optimization. Objectives generally include the minimization of cycle-time [38, 41] or energy [89, 90, 91]. Further heuristic objectives concern the manipulability of the robot [16, 44, 101, 112], the required effort, i.e. the squared velocity [48], acceleration, or torque along the path [4, 27, 82], and maximization of the stiffness of the manipulator [74]. In this chapter, no further contributions to this topic will be made, but several existing objectives will be taken into account.

The major field of application in existing papers on path placement is robot-based machining [74, 89, 90, 105]. Further publications exist which are independent of specific applications [4, 48, 82, 85]. This chapter will use robot-based thermal spray coating as a case study.

Most of the mentioned publications on path placement are devoted to manipulators without redundant axes. In this case, the trajectories can be constructed by inverse kinematics which leaves just the time-parametrization as degree of freedom for trajectory optimization. In this case, path placement is of particular importance for the quality of the resulting paths. Path placement with redundant axes is treated in [48, 85]. This chapter will focus on non-redundant manipulators, but it will also treat the problem in a way which is suited for the case of redundant manipulators.

The optimization methods applied to task and path placement are mostly regular sampling [16, 21, 103], nonlinear programming [41, 44, 48, 83], genetic algorithms [4, 74, 81, 82, 85, 90, 105], and simulated annealing [8]. A further approach is to use the response surface method combined with a local optimization method [56]. One aim of this chapter is to systematically investigate the performance of such methods for the path placement problem.

5.3 Problem Specification

The input of the problem of manipulator placement optimization considered in this chapter is a workpiece, a tool, a tool path, and a manipulator. The tool is mounted on the manipulator. The tool path is given relative to the workpiece so that it causes the desired effect on the workpiece surface according to the manufacturing process. The aim is to find a location, or pose, of the workpiece relative to the manipulator so that the manipulator is able to execute the tool path. A workpiece pose of this kind is called feasible. Additionally, the motion performed by the manipulator during execution of the tool path should be as favorable as possible. This is expressed by certain objectives. In the following, the details are specified.

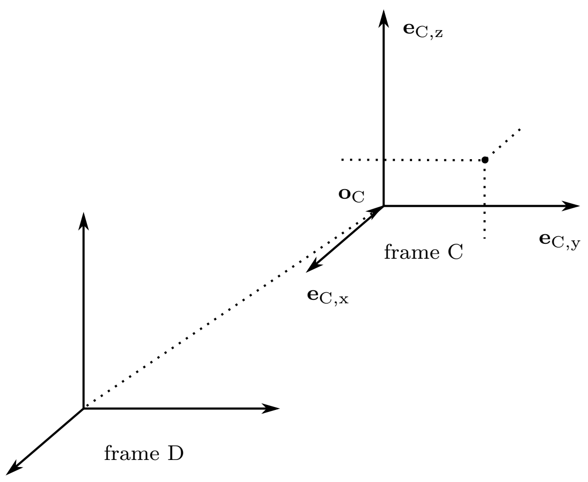

$\mathbf{e}_{\mathrm{C,z}}$
$\mathbf{o}_{\mathrm{C}}$
frame C
$\mathbf{e}_{\mathrm{C,y}}$
$\mathbf{e}_{\mathrm{C,x}}$
frame D

Fig. 5.1: Relation between frame C and frame D with regard to the coordinate transformation from C to D. The origin $\mathbf{o}_{\mathrm{C}}$ and the coordinate vectors of frame C are represented relative to frame D.

5.3.1 Reference Frames

The basic mechanism used for the problem specification is the reference frame, which specifies a position and orientation relative to another reference frame. A *reference frame* C can be represented by three orthonormal coordinate vectors $\mathbf{e}_{\mathrm{C,x}}$, $\mathbf{e}_{\mathrm{C,y}}$, $\mathbf{e}_{\mathrm{C,z}}$ corresponding to coordinates x_{c}, y_{c}, z_{c}, and an origin $\mathbf{o}_{\mathrm{C}}$ to which they are attached.

A reference frame C can be represented relative to a reference frame D by specifying its coordinate vectors by vectors in D, and its origin by a point in D. In this case,

$$^{\mathrm{D}}\mathbf{T}_{\mathrm{C}} = (\hat{\mathbf{e}}_{\mathrm{C,x}}\ \hat{\mathbf{e}}_{\mathrm{C,y}}\ \hat{\mathbf{e}}_{\mathrm{C,z}}\ \hat{\mathbf{o}}_{\mathrm{C}}) \tag{5.1}$$

defines a homogeneous coordinate transformation matrix from frame C to frame D (cf. Fig. 5.1). Interpreted as a rotation and translation when applied to a point, the point is modified according to the rotation and translation from D to C. $\hat{\mathbf{e}}_{\mathrm{C},i}$ emerges from $\mathbf{e}_{\mathrm{C},i}$, $i \in \{\mathrm{x,y,z}\}$, by adding a fourth component 0, and $\hat{\mathbf{o}}_{\mathrm{C}}$ from $\mathbf{o}_{\mathrm{C}}$ by adding a fourth component 1. The matrix $^{\mathrm{D}}\mathbf{R}_{\mathrm{C}} = (\mathbf{e}_{\mathrm{C,x}}\ \mathbf{e}_{\mathrm{C,y}}\ \mathbf{e}_{\mathrm{C,z}})$ describes the rotational component of the transformation and the vector $\mathbf{o}_{\mathrm{C}}$ its translational component.

Reference frames can also be represented by dual quaternions [57]. Moreover, the rotational part of the transformation $^{\mathrm{D}}\mathbf{T}_{\mathrm{C}}$ can alternatively be represented by a quaternion, Euler angles, or a rotation axis (Euler axis) and an angle [107]. A useful norm on rotations is the absolute value of the rotation angle, which can be calculated from quaternions (cf. Kuffner [65], Section V) or the axis-angle representation. A metric on orientations is based on the norm of the shortest rotation between two orientations. A norm on the space of transformations can be defined as the sum of this norm and the Euclidean norm on the space of translations.

5.3.2 Tool Paths, Manipulators, and Manipulator Paths

The production system is described with respect to a basic *world reference frame* W. The geometry of the workpiece is represented in a *workpiece (part) reference frame* P. The *location of the workpiece*, i.e. the position of P relative to W, is given by a transformation

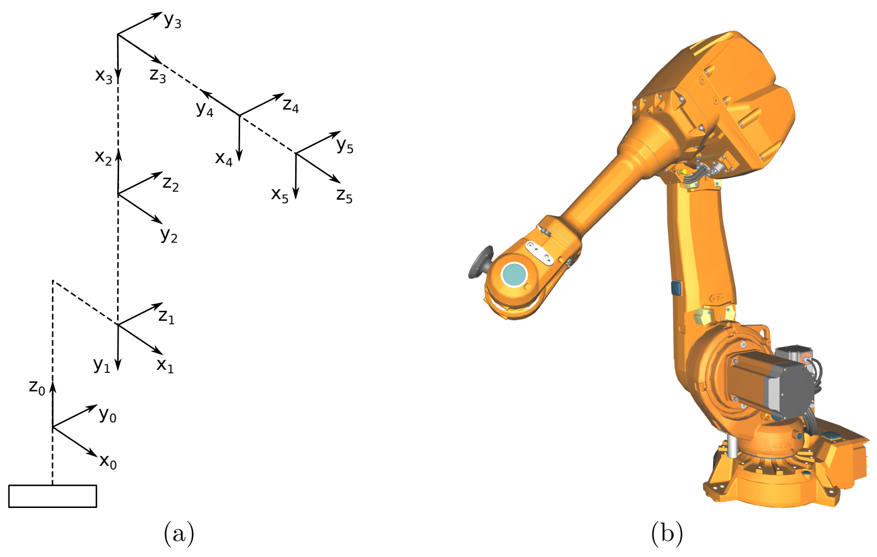


Fig. 5.2: (a) Kinematic chain of the ABB IRB 4600 industrial robot. Rotation is around the z-axis of each joint, i.e. $\mathbf{z}_0, \dots, \mathbf{z}_5$. (b) Geometric model of the ABB IRB 4600.

${}^{\mathrm{W}}\mathbf{T}_{\mathrm{P}}$. The geometry of the tool is represented in a *tool reference frame* T. A *tool path* is given by a time-dependent transformation ${}^{\mathrm{P}}\mathbf{T}_{\mathrm{T}}(t)$, which describes the location of the tool frame T relative to the workpiece frame P at time $t \in [0, t_f]$, $t_f > 0$.

The geometry of the manipulator is represented with respect to a *manipulator reference frame* M. The *location of the manipulator*, i.e. the position of M relative to the world frame W, is specified by a transformation ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{M}}$. In this chapter, the manipulator is assumed to have the form of an arm, i.e. a *linear kinematic chain of rigid segments connected by rotational joints*. The root of the arm is fixed to the manipulator frame M. At the other end of the kinematic chain the tool is fixed at the flange, which extends the chain to the so-called *end-effector* or *tool center point* (TCP). Many industrial robots like e.g. the ABB IRB 4600 used for evaluations in this chapter are of this type. In this case, the pose of the robot is determined by the values of the joint angles which together define a *configuration* of the robot, expressed by a *configuration vector* $\boldsymbol{\varphi}$. The ABB IRB 4600 has six joints with axial rotation. The configuration is a 6D-vector, whose entries are the rotational angles $\varphi_0, \dots, \varphi_5$ of the six axes. Figure 5.2 shows the geometric model and a diagram of the kinematic chain of the ABB IRB 4600. The rotation is around the z-axis of each joint. The location of the *end-effector reference frame* $\mathrm{E_M}$ is represented relative to the manipulator frame M by a transformation ${}^{\mathrm{M}}\mathbf{T}_{\mathrm{E_M}}$. A so-called *forward-kinematic calculation* allows to determine the location ${}^{\mathrm{M}}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi})$ of the end-effector for a given configuration $\boldsymbol{\varphi}$.

A *manipulator path* is given by a time-dependent sequence $\boldsymbol{\varphi}(t)$, $t \in [0, t_f]$, $t_f > 0$. The corresponding *end-effector path* is ${}^{\mathrm{M}}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi}(t))$, $t \in [0, t_f]$.

The execution of a tool path by the manipulator can be expressed by an equation demanding that the end-effector path is identical with the tool path, both represented relative to the world reference frame:

$$ {}^{\mathrm{W}}\mathbf{T}_{\mathrm{M}}{}^{\mathrm{M}}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi}(t)) = {}^{\mathrm{W}}\mathbf{T}_{\mathrm{P}}{}^{\mathrm{P}}\mathbf{T}_{\mathrm{T}}(t), \; t \in [0, t_f]. \tag{5.2} $$

5.3.3 System Control and Reaction

The manipulator, the tool, and the workpiece define a time-dependent system. At any time t, its state is described by a *state vector* $\mathbf{x}(t)$. The system behavior is influenced by a controller which yields a time-dependent *control vector* $\mathbf{u}(t)$. The system reaction to the control input is described by a differential equation which defines the change of the state vector over time depending on the state and control vector:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{5.3}$$

where $\dot{\mathbf{x}}$ denotes the derivative w.r.t. time. For systems employing manipulators with rotational joints,

$$\mathbf{x}(t) = (\boldsymbol{\varphi}(t), \boldsymbol{\omega}(t))^\top, \tag{5.4}$$

where $\boldsymbol{\omega}(t)$ is the vector of angular velocities of the joints.

The *system reaction* can be specified kinematics-based or dynamics-based. An example of a *dynamics-based formulation* is [40]

$$\begin{aligned} \mathbf{u}(t) &= \boldsymbol{\tau}(t), \\ \dot{\boldsymbol{\varphi}}(t) &= \boldsymbol{\omega}(t), \\ \dot{\boldsymbol{\omega}}(t) &= \mathbf{E}(\boldsymbol{\varphi}(t))^{-1}(\boldsymbol{\tau}(t) - \mathbf{C}(\boldsymbol{\varphi}(t), \boldsymbol{\omega}(t))\boldsymbol{\varphi}(t) - \boldsymbol{\tau}_\mathrm{g}(\boldsymbol{\varphi}(t)), \end{aligned} \tag{5.5}$$

where $\boldsymbol{\tau}(t)$ denotes the *vector of torques* of the joints, $\mathbf{E}$ is the inertia matrix of the manipulator, $\mathbf{C}$ represents the centrifugal and Coriolis forces and $\boldsymbol{\tau}_\mathrm{g}$ denotes the gravitation-induced torque. Friction has been omitted in this formulation.

An example of a *kinematics-based formulation* for manipulators with rotational joints is

$$\begin{aligned} \mathbf{u}(t) &= \boldsymbol{\alpha}(t), \\ \dot{\boldsymbol{\varphi}}(t) &= \boldsymbol{\omega}(t), \\ \dot{\boldsymbol{\omega}}(t) &= \boldsymbol{\alpha}(t), \end{aligned} \tag{5.6}$$

where $\boldsymbol{\alpha}(t)$ denotes the *vector of angular accelerations* of the joints.

5.3.4 Task Definition

The *objective to execute a time-dependent task* by the manipulator in an optimized way is specified as an optimal control problem:

$$\min_{\mathbf{u}} F(\mathbf{x}, \mathbf{u}) \tag{5.7}$$

subject to the system reaction

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{5.8}$$

subject to state and control constraints

$$\mathbf{h}_\mathrm{s}(\mathbf{x}, \mathbf{u}) \geq \mathbf{0}, \tag{5.9}$$

application constraints

$$\mathbf{h}_\mathrm{a}(\mathbf{x}) = \mathbf{0}, \tag{5.10}$$

and collision constraints

$$\mathbf{h}_{\mathrm{c}}(\mathbf{x}) > \mathbf{0}. \tag{5.11}$$

The *state and control constraints* restrict the joint angles, angular velocities, and angular accelerations (in the kinematics-based model, analogously for torques in the dynamics-based model) to intervals,

$$\mathbf{h}_{\mathrm{s}}(\mathbf{x},\mathbf{u}) = \begin{pmatrix} \boldsymbol{\varphi} - \boldsymbol{\varphi}^{\min} \\ \boldsymbol{\varphi}^{\max} - \boldsymbol{\varphi} \\ \boldsymbol{\omega}^{\max} - \boldsymbol{\omega} \\ \boldsymbol{\omega} + \boldsymbol{\omega}^{\max} \\ \boldsymbol{\alpha}^{\max} - \boldsymbol{\alpha} \\ \boldsymbol{\alpha} + \boldsymbol{\alpha}^{\max} \end{pmatrix}, \tag{5.12}$$

where $\boldsymbol{\varphi}^{\min}$ and $\boldsymbol{\varphi}^{\max}$ are the minimum and maximum joint angles and $\boldsymbol{\omega}^{\max}$ and $\boldsymbol{\alpha}^{\max}$ are the maximum joint angular velocities and accelerations. Limiting the derivative of the torque or acceleration may also be useful to produce a smooth motion.

There is one *application constraint* which serves as a coupling of the tool path ${}^{\mathrm{P}}\mathbf{T}_{\mathrm{T}}(t)$ and the manipulator path $\boldsymbol{\varphi}(t)$:

$$\mathbf{h}_{\mathrm{a}}(\mathbf{x}) = {}^{\mathrm{W}}\mathbf{T}_{\mathrm{M}}{}^{\mathrm{M}}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi}(t)) - {}^{\mathrm{W}}\mathbf{T}_{\mathrm{P}}{}^{\mathrm{P}}\mathbf{T}_{\mathrm{T}}(t),\ t \in [0, t_f], \tag{5.13}$$

where ${}^{\mathrm{P}}\mathbf{T}_{\mathrm{T}}(t)$ is the given tool path.

Collision constraints serve to avoid mutual collisions between parts of the manipulator, the tool, the workpiece, and a potential environment. The items involved are rigid geometric objects. In this case, collision avoidance can be formulated by demanding that the minimal distance between each relevant pair of objects is greater than zero,

$$\mathbf{h}_{\mathrm{c}}(\mathbf{x}) = d(\boldsymbol{\varphi}(t)), \tag{5.14}$$

where $d(\boldsymbol{\varphi}(t))$ is the minimum over all minimum distances of relevant pairs of objects for the geometric manipulator pose induced by the joint angles $\boldsymbol{\varphi}(t)$. The fulfillment of the constraint can be tested algorithmically by existing efficient collision detection algorithms [71]. For the experimental evaluation of the chapter, objects in boundary representation and the algorithm by Larsen et al. [66] have been used, which is also capable of calculating the minimum distance between objects. An alternative collision constraint would be to demand that the intersection of each pair of objects is the empty set.

The *objective function* is

$$F(\mathbf{x},\mathbf{u}) = \int_0^{t_f} L(\mathbf{x}(t),\mathbf{u}(t))\,\mathrm{d}t, \tag{5.15}$$

where $\mathbf{x}$ and $\mathbf{u}$ are defined as before. The kernel L is a weighted sum of several sub-objectives, i.e.

$$L = \lambda_\alpha \cdot L_\alpha + \lambda_{\mathrm{E}} \cdot L_{\mathrm{E}} + \lambda_{\mathrm{m}} \cdot L_{\mathrm{m}} + \lambda_{\mathrm{c}} \cdot L_{\mathrm{c}} + \lambda_{\mathrm{col}} \cdot L_{\mathrm{col}} + \lambda_{\mathrm{l}} \cdot L_{\mathrm{l}}\,, \tag{5.16}$$

with non-negative weights $\lambda_{\cdot}$.

The term

$$L_\alpha = \|\boldsymbol{\alpha}(t)\|^2 \tag{5.17}$$

concerns the *minimization of control* in the kinematics case. In the dynamics case, $\boldsymbol{\alpha}$ is replaced with $\boldsymbol{\tau}$. It serves for regularization and for the reduction of the wear of the manipulator. Alternatively, the angular velocity could be minimized,

$$L_\omega = \|\boldsymbol{\omega}(t)\|^2. \tag{5.18}$$

The term

$$L_\mathrm{E} = \sum_{i=0}^{d} (\tau_i(t)\dot{\varphi}_i(t))^2, \tag{5.19}$$

$d > 0$ the number of joints, is related to the *minimization of the rotation energy*

$$E_i = \int_0^{t_f} \tau_i(t)\dot{\varphi}_i(t)\,\mathrm{d}t$$

at every joint i.

The terms L_m and L_c are related to the *optimization of manipulability.* They are based on the Jacobian matrix $\mathbf{J}(\boldsymbol{\varphi})$ which transforms the rotational joint velocities into translational and rotational velocities $\mathbf{v}$ in the manipulator frame by $\mathbf{v} = \mathbf{J}(\boldsymbol{\varphi})\dot{\boldsymbol{\varphi}}$. The i-th column of the Jacobian matrix for a manipulator arm with rotational joints is calculated as [112]

$$\mathbf{J}_i = \begin{pmatrix} \mathbf{e}_{\mathrm{z},i} \times (\mathbf{o}_E - \mathbf{o}_i) \\ \mathbf{e}_{\mathrm{z},i} \end{pmatrix},$$

where $\mathbf{e}_{\mathrm{z},i}$ is the i-th joint axis, $\mathbf{o}_E$ is the end effector position and $\mathbf{o}_i$ is a origin of the i-th joint axis. Yoshikawa [112] defined a *manipulability measure*

$$w_\mathrm{m}(\boldsymbol{\varphi}) = \sqrt{\det(\mathbf{J}(\boldsymbol{\varphi})\mathbf{J}^\top(\boldsymbol{\varphi}))},$$

or for non-redundant manipulators, which have a symmetric Jacobian matrix, as [112]

$$w_\mathrm{m}(\boldsymbol{\varphi}) = |\det(\mathbf{J}(\boldsymbol{\varphi}))|.$$

Togai [100] proposes a *condition number*

$$w_\mathrm{c}(\boldsymbol{\varphi}) = \frac{1}{\|\mathbf{J}(\boldsymbol{\varphi})\|\|\mathbf{J}^{-1}(\boldsymbol{\varphi})\|} = \frac{\sigma_\mathrm{min}}{\sigma_\mathrm{max}}$$

in the interval $[0, 1]$, where σ_min and σ_max are the minimum and maximum singular value of $\mathbf{J}$.

The manipulability measure and the condition number are used for demanding *maximization of manipulability* by defining

$$L_\mathrm{m} = \frac{1}{w_\mathrm{m}(\boldsymbol{\varphi}(t)) + \varepsilon} \text{ and } L_\mathrm{c} = \frac{1}{w_\mathrm{c}(\boldsymbol{\varphi}(t)) + \varepsilon}. \tag{5.20}$$

This definition transfers maximization to minimization and avoids singularities. A recommended choice of ε is $\varepsilon = 1 \cdot 10^{-6}$.

In addition to the collision avoidance constraint, the intention of the *objective of collision avoidance* L_col is to maximize the distance between potentially colliding parts. The corresponding definition uses a logarithmic barrier function,

$$L_\mathrm{col} = -\log(d(\boldsymbol{\varphi}(t)) + \varepsilon), \tag{5.21}$$

with $d(\boldsymbol{q}(t))$ as defined before. A recommended choice of ε is $\varepsilon = 1 \cdot 10^{-10}$. The objective may also be used as a soft constraint of collision avoidance by omitting the collision avoidance constraint.

Similarly, the *objective of joint limit avoidance* L_l intends to keep configurations away from the joint limits in order to prevent a potential violation of these limits for small changes caused by inaccuracies. To achieve this, logarithmic barrier functions are also used:

$$w_\text{l}(\boldsymbol{\varphi}) = \sum_{i=0}^{d} \Bigg[-c_i \log\left(\frac{\varphi_i^\text{max} - \varphi_i}{\varphi_i^\text{max} - \varphi_i^\text{des}} + \varepsilon\right) \cdot \left(\frac{\varphi_i^\text{max} - \varphi_i^\text{des}}{\varphi_i^\text{des} - \varphi_i^\text{min}}\right) \\ - c_i \log\left(\frac{\varphi_i - \varphi_i^\text{min}}{\varphi_i^\text{des} - \varphi_i^\text{min}} + \varepsilon\right)\Bigg], \tag{5.22}$$

with the desired angle $\varphi_i^\text{des} \in [\varphi_i^\text{min}, \varphi_i^\text{max}]$, weights $c_i > 0$, and a recommended $\varepsilon = 1 \cdot 10^{-10}$, which results in the objective

$$L_\text{l} = w_\text{l}(\boldsymbol{\varphi}(t)). \tag{5.23}$$

5.3.5 Manipulator and Workpiece Placement

The placement of the manipulator is specified by the relative location ${}^\text{W}\mathbf{T}_\text{M}$ of the manipulator frame M in the world frame W. Analogously, the placement of the workpiece is defined by the relative location ${}^\text{W}\mathbf{T}_\text{P}$ of the workpiece frame P in the world frame W. If no further environment is present, as is the case for the investigations of this chapter, it is sufficient to place just one of those two items and keep the other one fixed, or use the frame ${}^\text{M}\mathbf{T}_\text{P}$ as variable. This leads to two equivalent placement problems, $\text{X} \in \{\text{P}, \text{M}\}$, which have the form of an optimization problem with constraints:

$$\min_{{}^\text{W}\mathbf{T}_\text{X}} \min_{\mathbf{u}} F({}^\text{W}\mathbf{T}_\text{X}, \mathbf{x}, \mathbf{u}) \tag{5.24}$$

subject to

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)),\ t \in [0, t_f],\ t_f > 0, \\ \mathbf{h}_\text{s}({}^\text{W}\mathbf{T}_\text{X}, \mathbf{x}, \mathbf{u}) &\geq \mathbf{0}, \\ \mathbf{h}_\text{a}({}^\text{W}\mathbf{T}_\text{X}, \mathbf{x}) &= \mathbf{0}, \\ \mathbf{h}_\text{c}({}^\text{W}\mathbf{T}_\text{X}, \mathbf{x}) &> \mathbf{0}, \\ \mathbf{h}_\text{p}({}^\text{W}\mathbf{T}_\text{X}) &\geq \mathbf{0}. \end{aligned}$$

Differences to the optimization problem of the preceding section are the varying location of the workpiece and the additional placement constraint $\mathbf{h}_\text{p}$. The latter can be used to restrict the allowed placements. Examples are to restrict the placement of the workpiece to a plane in parallel to the x-y-plane or to restrict the potential rotations of workpiece in order to model the restrictions of the fixture of the workpiece.

5.4 Overview of the Solution

The optimization problem (5.24) has the form of two nested optimization problems, *placement optimization* and *manipulator path optimization*. The domains over which the optimizations of the two problems have to be performed are quite different. Placement optimization takes place over the domain of workpiece or manipulator placements which has six real-valued degrees of freedom. The domain of the manipulator path optimization consists of control functions which are multi-valued functions with the time as parameter, which have in general infinitely many degrees of freedom. A common approach to reduce the complexity is to replace the general function space with a function space of finite dimension. The application to control functions leads to

$$\mathbf{u}(t) = \sum_{i=0}^{n} \mathbf{u}_i \cdot M_i(t),\ t \in [0, t_f], \tag{5.25}$$

where $M_i(t)$ are suitable basis functions and $\mathbf{u}_i$ are control parameters, $i = 0, \ldots, n$. The aim is to have a small number n since it determines the dimension of the optimization problem. The resulting parameter optimization problem to find a suitable number n may be solved by intuition, as done in this chapter, or systematically.

Restricting the domain of functions in this way may cause the problem that some constraints cannot be fulfilled. Particularly critical are constraints demanding equality. In the path optimization problem considered here, the application constraint (5.13) has this type. It might happen that the given tool path cannot be represented by the resulting function type. A possibility to cope with this problem is to replace critical equality constraints by objective functions. Doing so for the application constraint (5.13) leads, according to (5.15), to an objective function $L_\mathrm{a}(\mathbf{u}_0, \ldots, \mathbf{u}_n, t)$ defined by the kernel

$$L_\mathrm{a}(\mathbf{u}_0, \ldots, \mathbf{u}_n, t) = \left\| {}^\mathrm{W}\mathbf{T}_\mathrm{M} {}^\mathrm{M}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi}(\mathbf{u}_0, \ldots, \mathbf{u}_n, t)) - {}^\mathrm{W}\mathbf{T}_\mathrm{P} {}^\mathrm{P}\mathbf{T}_\mathrm{T}(t) \right\|^2, \tag{5.26}$$

where the norm described in Section 5.3.1 may be taken, and the optimum is searched for over the parameters $\mathbf{u}_i$. A necessary condition for the fulfillment of the constraint is that the partial derivatives are equal to 0, i.e.

$$\frac{\partial L_\mathrm{a}(\mathbf{u}_0, \ldots, \mathbf{u}_n, t)}{\partial \mathbf{u}_i} = 0,\ i = 0, \ldots, n,\ t \in [0, t_f], \tag{5.27}$$

Those constraints may replace the objective (5.26).

Alternatively, the approximative version

$$F_\mathrm{a}(\mathbf{u}_0, \ldots, \mathbf{u}_n) = \int_0^{t_f} L_\mathrm{a}(\mathbf{u}_0, \ldots, \mathbf{u}_n, t)\,\mathrm{d}t$$

of the constraint may be added as a penalty term to the objective function of the original problem. This is in fact a sort of relaxation of the original problem which will be further considered in Section 5.9.2.

A further issue is the representation of the tool path ${}^\mathrm{P}\mathbf{T}_\mathrm{T}(t)$. A possibility to become independent of a specific representation is to represent the tool path by a sequence of gun configurations taken at sampling times $t_0, \ldots, t_m$, $t_0 = 0$, $t_m = t_f$. The methods for tool path planning in Section 3.6 represent a tool path by a sequence of gun reference frames each tagged with a spray time density. With Eq. (2.9), the spray time densities can be replaced with spray times from which the sampling times are obtained as partial sums.

Taking the t_i as domain decomposition for an approximation of the integral by a Riemann sum leads to a representation of the objective function F_a by a finite sum

$$F_\mathrm{a}(\mathbf{u}_0, \ldots, \mathbf{u}_n) \approx \sum_{j=0}^{m} \left\| {}^\mathrm{W}\mathbf{T}_\mathrm{M} {}^\mathrm{M}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi}(\mathbf{u}_0, \ldots, \mathbf{u}_n, t_j)) - {}^\mathrm{W}\mathbf{T}_\mathrm{P} {}^\mathrm{P}\mathbf{T}_\mathrm{T}(t_j) \right\|^2 \cdot \Delta t_j\,, \tag{5.28}$$

with

$$\Delta t_0 = \frac{t_1 - t_0}{2},\ \Delta t_j = \frac{t_{j+1} - t_{j-1}}{2},\ j = 1, \ldots, m-1,\ \Delta t_m = \frac{t_m - t_{m-1}}{2}\,. \tag{5.29}$$

The sequence $\boldsymbol{\varphi}(\mathbf{u}_0, \ldots, \mathbf{u}_n, t_j)$, $j = 1, \ldots, m$, represents the manipulator path by sampled angular configurations, analogously to the tool path.

An alternative view is to consider the sampling representation as a piecewise constant representation of a tool path with basis functions $M_j(t)$ defined as

$$M_j(t) := \begin{cases} 1, & \text{if } t \in [(t_{\max(0,j-1)} + t_j)/2, (t_j + t_{\min(m,j+1)})/2], \\ 0, & \text{otherwise.} \end{cases}$$

Using this representation for the tool path and for the control function (5.25) makes the approximation in Eq. (5.28) exact.

An important special case of the path placement problem is that of manipulators without redundant degrees of freedom. The ABB IRB 4600 from Section 5.3.2 has this property. For such manipulators at most a finite number of poses exist for every given end-effector position, except for the so-called singular positions. Those poses can be determined by the inverse kinematics calculation. For a fixed placement of the tool path relative to the robot, this implies an at most finite number of manipulator paths that induce a given tool path. These manipulator paths are obtained by selecting the potential poses at the start of the tool path and tracking them along the tool path. In the discrete case of Eq. (5.28), the discrete tracking of the poses derived from the tool configurations ${}^{\mathrm{P}}\mathbf{T}_{\mathrm{T}}(t_i)$ by inverse kinematics leads to finitely many sequences Φ of configuration vectors $\boldsymbol{\varphi}_{\mathrm{t},j}$, $j = 0, \ldots, m$. Manipulator paths which have such a configuration sequence as sampling sequence fulfill the approximate constraint (5.28) perfectly with optimum value 0.

For control functions of the type (5.25), manipulator paths corresponding to the configuration sequences Φ are now obtained from the following version of (5.28):

$$F_{\mathrm{a}}(\mathbf{u}_0, \ldots, \mathbf{u}_n) \approx \sum_{j=0}^{m} \|\boldsymbol{\varphi}(\mathbf{u}_0, \ldots, \mathbf{u}_n, t_j) - \boldsymbol{\varphi}_{\mathrm{t},j}\|^2 \cdot \Delta t_j\,. \tag{5.30}$$

For the kinematic case, which is considered here, the angular accelerations of the joints have been proposed as control function according to Eq. (5.6). Motivated by the tight geometric coupling of the path to the samples, it seems sufficient to replace the acceleration-based control by a position-based, geometric control, i.e.

$$\boldsymbol{\varphi}(\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_n, t) = \sum_{i=0}^{n} \boldsymbol{\varphi}_i \cdot M_i(t),\ t \in [0, t_f], \tag{5.31}$$

where the $\boldsymbol{\varphi}_i$, $i = 0, \ldots, n$, now define the domain of optimzation. By this simplification, the ability to change the velocity and acceleration along the path without a reparametrization is lost, which is acceptable for path placement.

When a higher order than the piecewise constant representation is employed in this chapter, e.g. to reduce the size of the optimization problem, multi-dimensional B-spline curves are used as representation (5.31), which usually induce fair curves.

Together with the change of the control function,

$$\hat{\mathbf{x}} = \boldsymbol{\varphi} \tag{5.32}$$

as state vector is also sufficient. This means that in fact the following simplified version of the placement problems (5.24) is considered:

$$\min_{{}^{\mathrm{W}}\mathbf{T}_{\mathrm{X}}} \min_{\hat{\mathbf{x}}} F({}^{\mathrm{W}}\mathbf{T}_{\mathrm{X}}, \hat{\mathbf{x}}) \tag{5.33}$$

subject to

$$\begin{aligned} \mathbf{h}_{\mathrm{s}}({}^{\mathrm{W}}\mathbf{T}_{\mathrm{X}}, \hat{\mathbf{x}}) &\geq \mathbf{0}, \\ \mathbf{h}_{\mathrm{a}}({}^{\mathrm{W}}\mathbf{T}_{\mathrm{X}}, \hat{\mathbf{x}}) &= \mathbf{0}, \\ \mathbf{h}_{\mathrm{c}}({}^{\mathrm{W}}\mathbf{T}_{\mathrm{X}}, \hat{\mathbf{x}}) &> \mathbf{0}, \\ \mathbf{h}_{\mathrm{p}}({}^{\mathrm{W}}\mathbf{T}_{\mathrm{X}}) &\geq \mathbf{0}, \end{aligned}$$

$\mathrm{X} \in \{\mathrm{P}, \mathrm{M}\}$.

This analysis immediately leads to a solution approach of separated placement and path optimization. This means that the optimization of the "outer problem", placement optimization, is performed based on solutions of the "inner problem", path optimization. The solution of the inner problem consists of selecting the best among the finitely many manipulator paths obtained by inverse kinematics.

The remainder of the chapter is devoted to the analysis of the performance of a number of methods which implement the separated approach. The performance analysis concerns the quality of the result and the computation time. It is performed empirically with the ABB IRB 4600 employed as manipulator in case studies of robot-based thermal spray coating [30]. In the case studies, a path of a spray gun over a workpiece surface is given whose execution achieves a desired coating height. The ABB IRB 4600 has a reach of about 2.05 m. Workpieces of two different sizes, "small" ($780 \times 850 \times 100$ mm) and "large" ($1200 \times 1300 \times 154$ mm), with precalculated spray gun paths are used. The reason for the different sizes is to analyze the influence of the size on the structure of the configuration space. The workpieces resemble the hood of a car at different sizes. The corresponding tool paths cover the whole surface. In both cases, the tool path consists of 3714 data points for the tool frame relative to the workpiece frame. Figures 5.15 and 5.16 give impressions of the scenarios. The timings were taken on an Intel Core i7-2600 at 3.4GHz with 16GB RAM. Parallelization on the four cores by OpenMP has been used for sampling and for the evolutionary algorithms.

First, a number of methods for separated placement and path optimization, divided in systematic sampling (Section 5.5), evolutionary approaches (Section 5.6), and local descent (Section 5.7) are analyzed. The intention is to get insight into their performance and suitability to the problem.

The tool path is given by 3714 gun configuration samples taken at a sequence of sampling times. The manipulator paths are represented by configuration samples at the same times.

The objective function is restricted to three sub-objectives,

$$L = \lambda_{\mathrm{l}} \cdot L_{\mathrm{l}} + \lambda_{\alpha} \cdot L_{\alpha} + \lambda_{\mathrm{c}} \cdot L_{\mathrm{c}}\,, \tag{5.34}$$

with $\lambda_{\mathrm{l}} = 0.0005$, $\lambda_{\alpha} = 0.002$, and $\lambda_{\mathrm{c}} = 5 \cdot 10^{-8}$. The reason is that L_{m} and L_{c} have a similar effect so that one of them has been omitted. The same holds for L_{α} and L_{ω}, too. L_{E} is not considered because it is related to dynamics. The weights are chosen such that the objectives are scaled to the same magnitude. The weights in L_{l} according to (5.22) are chosen as $c_0 = 0$ and $c_i = 1$ otherwise.

Additionally, an objective resulting from the conversion of the hard collision constraint is considered. In contrast to (5.21), the number of time samples t_j, $j = 0, \dots, m$, at which a collision occurs is taken as objective function.

Furthermore, two additional placement constraints are employed. The first one is that the workpiece has to be above the base plane of the robot, and the second one that the workpiece surface should face to the upper hemisphere.

The constraints are tested during the sampling, except for the acceleration constraint, which is already covered by the acceleration minimization objective. The additional application constraints are applied by projecting the solution back in the case of a violation. The projection is performed by rotating the manipulator in the shortest way back to the feasible space and by translating the manipulator along the z-axis such that the workpiece is above the base plane of the manipulator.

Furthermore, as an alternative approach, two simultaneous solutions with relaxation are presented in Section 5.8 and 5.9. The idea is to consider the workpiece as movable over time or to allow deviations from the predefined tool path, thus extending the search space in the hope to simplify its structure. The resulting problem is optimized over control functions, incorporating solution methods also applicable to redundant manipulators.

5.5 Systematic Sampling

In the following, systematic sampling is described for the case of manipulator placement optimization, i.e. the version $\mathrm{X} = \mathrm{M}$ of (5.24). Approaches of systematic sampling consist of four major tasks:

1. Search space reduction.
2. Systematic choice of samples in the reduced search space.
3. Feasibility checking for every chosen sample.
4. Optimized manipulator path calculation for every feasible sample.

A common approach to *search space reduction* is to take into account the reach of the manipulator in relation to the tool locations on the tool path (Section 5.5.1).

The *choice of samples* takes place over the manipulator locations ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{M}}$. They are composed of a translational and a rotational part. The translations $\mathbf{o}_{\mathrm{M}}$ are considered as points in 3D-space so that point samples of a domain in space are taken. The rotational part is represented by the z-axis vector $\mathbf{e}_{\mathrm{M,z}}$, and a rotation around the axis induced by this vector. For the ABB IRB 4600, the latter rotation can be set to an arbitrary constant value since its root axis offers this degree of freedom as well. The vectors $\mathbf{e}_{\mathrm{M,z}}$ are considered as points on a unit sphere from which point samples are taken. Sections 5.5.2 and 5.5.3 describe two sampling strategies over this representation.

Feasibility checking is performed by an inverse kinematics calculation for every data point of the tool path according to Section 5.4. Every resulting manipulator pose is checked for collision. If a collision-free manipulator pose is found for every tool path data point the manipulator placement sample is considered as feasible. To improve the runtime, the tool path data points are not processed sequentially, but in a different permutation by employing the van der Corput sequence [29]. In this way, the evaluation might abort earlier in the case of infeasibility.

The *calculation of an optimized manipulator path* of a feasible sample is performed according to Section 5.4. Different paths are composed sequentially by starting with the manipulator configurations for the first tool path data point, and continuing with the next configuration closest to the current configuration. The path with the best score according to the objective function is returned as the result.

5.5.1 Search Space Reduction by Reachability Constraints

A necessary condition of feasibility of a manipulator position is the reachability of all tool positions of the given tool path. Three heuristics are presented in the following which make use of this observation. They refer to the translational component. The calculations are performed in the world reference frame.

The first approach is the calculation of a so-called *distance ball*. The radius of the distance ball is the maximum possible extension of the manipulator, i.e. the maximum possible distance between the root of the manipulator and the tool center point of a tool attached to its end-effector. The origin of the tool frame is taken as tool center point. A feasible region for the location of the root of the manipulator is obtained by taking the tool center of one of the tool path data points as center of the ball.

Evidently, an immediate improvement can be achieved by the approach of *distance ball intersection*. Here, a distance ball with the same radius is located at several, or all, tool path data points, and the intersection of all those balls is taken as feasible region. A possible representation of the result is a regular spatial grid of suitable extension in which vertices are labeled as feasible or infeasible. The decision at a vertex can be taken by calculating its distance to all ball centers. If the maximum distance is at most the maximum extension of the manipulator, the vertex is feasible.

This approach can be further refined by taking the *maximum of obstacle-aware distance functions*. In contrast to the distance ball intersection, the distance calculations to the centers of the selected tool path data points take obstacles into account. The distance function of one of the centers is a function in space which assigns to every point in space the length of a shortest free-space path to the center. In the investigations here, the only obstacle considered is the workpiece. A point in space is considered feasible if the maximum of all distance values is at most the maximum extension of the manipulator. The approach is feasible since the manipulator arm can also be considered as path between its root and the tool, although usually not the shortest one.

The feasible domain can again be calculated and represented over a regular grid. In this case, the distance functions can be calculated using the second-order fast marching method [92]. Obstacles are incorporated by removing edges from the grid. The boundaries of the feasible regions in the grid-based representation can e.g. be extracted and visualized with the marching cubes algorithm [77].

Figure 5.3 shows the effect of the different heuristics of search space reduction for the workpiece and the ABB IRB 4600. The distance ball shown in Fig. 5.3(a) is placed at a tool path data point close to the center of all tool path data points. The example of distance ball intersection of Fig. 5.3(b) is taken over 100 balls whose centers are uniformly distributed over the tool path. The example with obstacle-aware distance functions in Fig. 5.3(c) uses the same tool path data points. The distance function calculation for the search space reduction was performed on a regular grid of $36 \times 36 \times 30$ vertices, which corresponds to a cell edge length of 200 mm. The total runtime of all distance function calculations using the fast marching method was approximately 1 second.

For the experimental investigations of the chapter, the approach of obstacle-aware distance functions has been used.

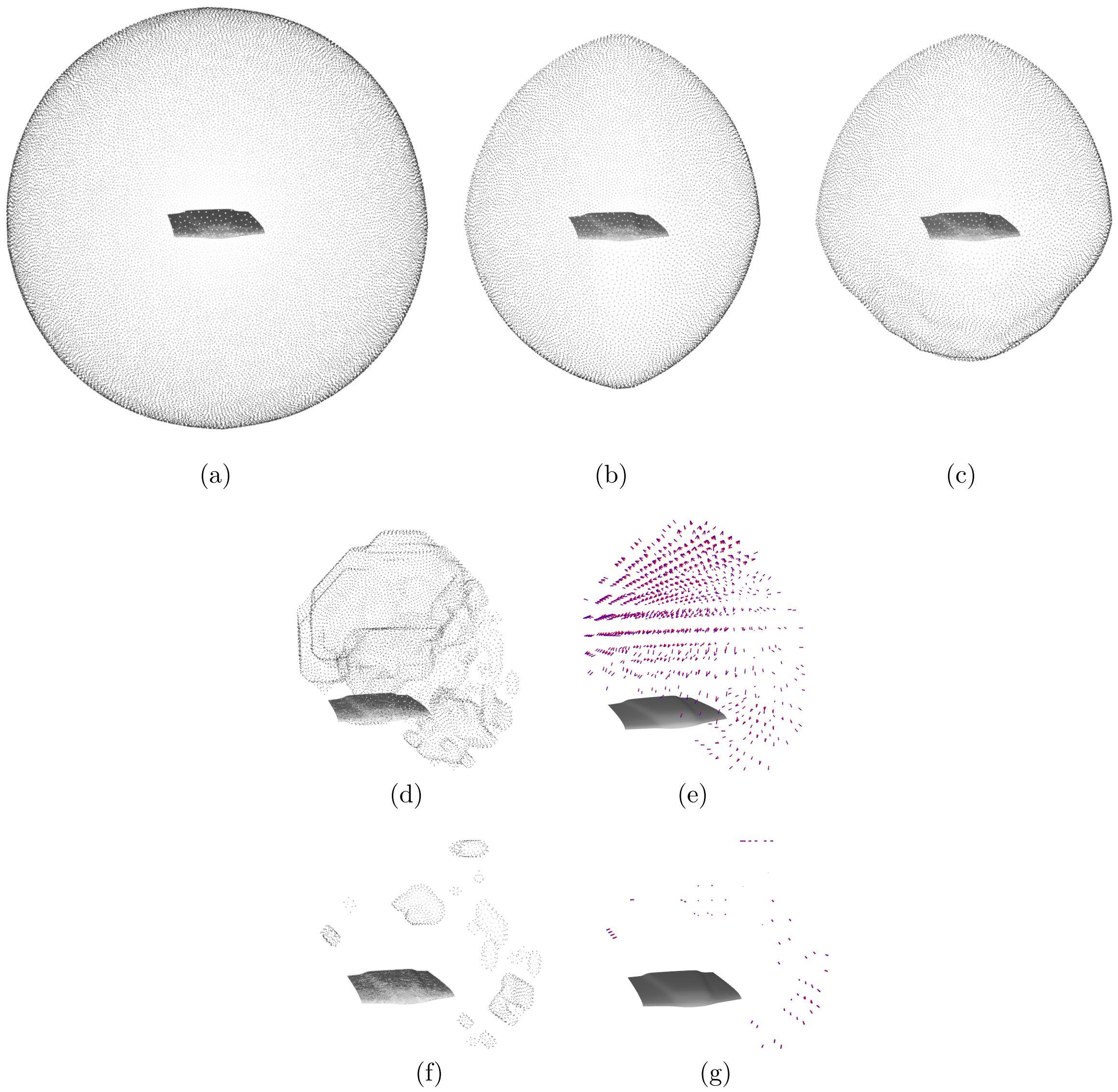


Fig. 5.3: Search space restriction according to the distance criteria: (a) distance ball with the radius of the length of the manipulator, (b) distance ball intersection, (c) maximum of obstacle-aware distance function, and the feasible domains from regular sampling: (d) solution domain, (e) solutions shown as vectors representing the z_0-axis of the manipulator, (f) solution domain with consideration of the additional application constraints, (g) the corresponding solutions shown as vectors.

Table 5.1: Optimization results for the small workpiece. The fitness is the value of the combined objective function.

Method	Timing [s]	Fitness	Remarks
Regular sampling	613	1.23064	5151 position, 48 rotation samples
LHS	10	1.47104	5000 samples
MO-CMA-ES rs	550	1.17242	solution population in Fig. 5.4
MO-CMA-ES lhs	635	1.13019	solution population in Fig. 5.5
CMA-ES	581	1.11498	initial deviation of 50 mm
$(\mu + \lambda)$-ES	348	1.11082	(25+50)-ES, 50 generations
Nelder-Mead	564	1.10864	initial distance of 50 mm
Gradient descent	264	1.27707	gradient by finite differences
Gradient descent	249	1.19563	gradient from the response surface

5.5.2 Regular Sampling

As described at the beginning of Section 5.5, the domain of the manipulator positions is a subset of the direct product of the 3D-space, representing translations, with a unit sphere, representing rotations. The third angle is not considered because of the rotational basis joint of the IRB 4600. The 3D-space is restricted by one of the heuristics of Section 5.5.1 to a finite region. The region is sampled on a regular 3D-grid defined by its axes-parallel rectangular bounding volume. The sphere is sampled using HEALPix [42]. HEALPix is a regular equal-area sampling on a 2-sphere, where the samples are also aligned to a number of lines of constant latitude.

In the following investigations, a $36 \times 36 \times 30$ grid for translations, and 48 rotational samples have been used. Figure 5.3(d) visualizes the feasible translational domain. Figure 5.3(e) additionally shows the feasible rotational components represented by short vectors. Figures 5.3(f) and (g) take the placement constrains, cf. Section 5.4, into account.

The sub-objective functions combined define a multidimensional function. Figures 5.4 and 5.7 show its range for the small and the large workpiece, respectively, visualized by different 2D-projections. The Pareto-optimal samples are additionally highlighted. A sample is Pareto-optimal, if no other sample exists with at least one better entry and no worse entries. The projection onto the L_l-L_c-plane shows that these two sub-objectives seem to be conflicting, i.e. that they cannot be minimized simultaneously. For the larger workpiece, considerably less feasible solutions exist because of reachability and collision issues.

The first entry of Tables 5.1 and 5.2 shows the calculation times and the value achieved for the combined objective function (fitness).

5.5.3 Sparse Sampling

A well-known method of sparse sampling is the *Latin hypercube sampling* (LHS) [78]. The region to be sampled is enveloped in a regular bounding volume. The bounding volume is divided by a regular grid into cells with indices in $\{0, \ldots, n-1\}^d$, d the dimension of the space to be sampled. Furthermore, a $n \times d$-matrix $\mathbf{X}$ is generated whose columns $\mathbf{X}_{\cdot,j}$ consist of random permutations of the numbers $0, \ldots, n-1$. Then n grid cells are selected whose indices are formed by the n rows of $\mathbf{X}$. In this way, no two cells have a same index entry in any dimension which intuitively means that the dimensions of the domain are maximally covered. From the selected cells, n samples are determined as a randomly chosen point in every cell.

Table 5.2: Optimization results for the large workpiece.

Method	Timing [s]	Fitness	Remarks
Regular sampling	18	2.14932	3380 position, 48 rotation samples
LHS	3	2.31818	5000 samples
MO-CMA-ES rs	420	2.06201	solution population in Fig. 5.7
MO-CMA-ES lhs	320	2.03039	solution population in Fig. 5.8
CMA-ES	458	2.01603	initial deviation of 50 mm
$(\mu + \lambda)$-ES	118	2.01894	(25+50)-ES, 50 generations
Nelder-Mead	503	2.01449	initial distance of 50 mm
Gradient descent	641	2.04475	gradient by finite differences
Gradient descent	470	2.04384	gradient from the response surface

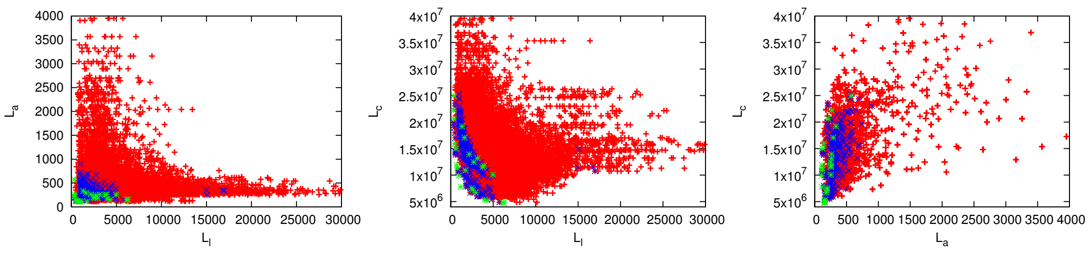


Fig. 5.4: Objective function values from regular sampling of Section 5.5.2 (red), its Pareto front (blue), and the solution population from MO-CMA-ES of Section 5.6 (green) for the smaller workpiece.

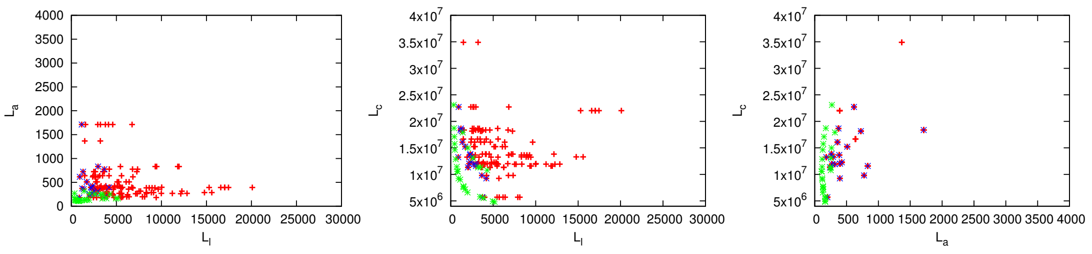


Fig. 5.5: Objective function values from LHS of Section 5.5.3 (red) and its Pareto front (blue), and the solution population from MO-CMA-ES of Section 5.6 (green) for the smaller workpiece.

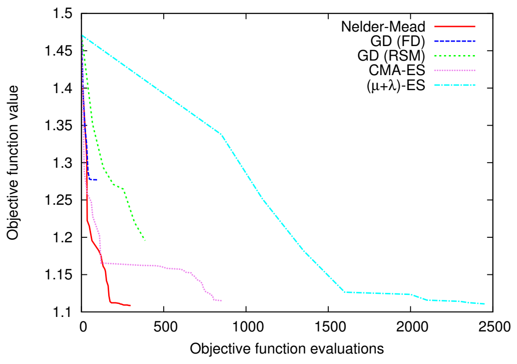


Fig. 5.6: Objective function values for the single objective optimization methods of Sections 5.6 and 5.7 for the smaller workpiece.

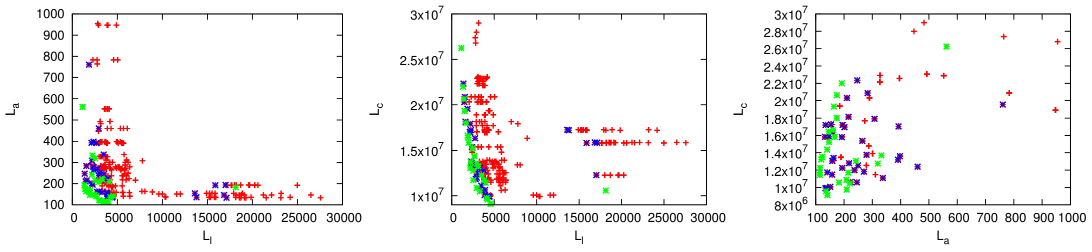


Fig. 5.7: Objective function values from regular sampling of Section 5.5.2 (red), its Pareto front (blue), and the solution population from MO-CMA-ES of Section 5.6 (green) for the larger workpiece.

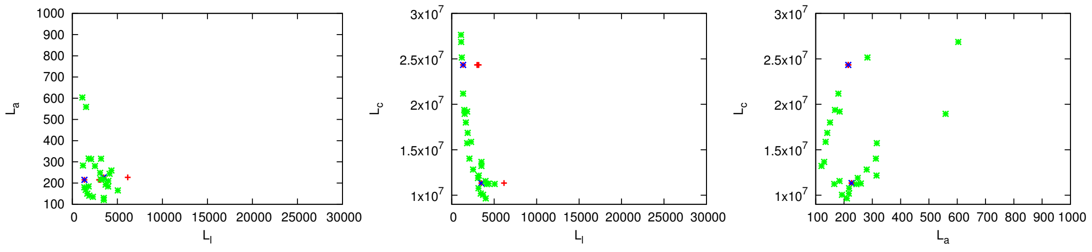


Fig. 5.8: Objective function values from LHS of Section 5.5.3 (red) and its Pareto front (blue), and the solution population from MO-CMA-ES of Section 5.6 (green) for the larger workpiece.

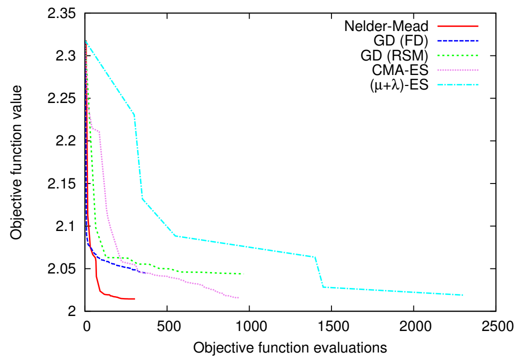


Fig. 5.9: Objective function values for the single objective optimization methods of Sections 5.6 and 5.7 for the larger workpiece.

Figures 5.5 and 5.8 visualize the range of the objective function values as projections of the sample vector of sub-objectives for the small and the large workpiece, respectively. The LHS has been only applied to the translational part, while the rotational part of every sample has been chosen randomly with uniform distribution. Among the originally about 5000 samples, only a few feasible solutions have been found.

The second entry of Tables 5.1 and 5.2 shows the calculation times and the value achieved for the combined objective function (fitness). LHS shows worse results than regular sampling, but requires considerably less running time. Both methods are often used for the initialization of other optimization approaches, as will also be done in the following.

5.6 Evolutionary Approaches

The basic principle of evolutionary algorithms is to vary a population of candidate solutions, represented by so-called individuals, by random mutation of an individual or recombination of several individuals, similar to the biological evolution. A popular evolutionary algorithm for single objective optimization of real valued functions is the *CMA-ES* [45]. It uses a self-adapting mutation operator which modifies the covariance matrix of a multi-variate normal distribution.

A multi-objective variant of the CMA-ES is the *MO-CMA-ES* [51, 53, 97, 106]. The MO-CMA-ES combines the covariance matrix adaptation mutation strategy [45] with non-dominated sorting and the contributing hypervolume as selection strategies to form a multi-objective optimization algorithm. Non-dominated sorting partitions the population into successive Pareto fronts, while the hypervolume-indicator is used to rank subsets of individuals in the same Pareto front according to the hypervolume they contribute.

Another similar algorithm is the $(\mu + \lambda)$-*evolution strategy* [10]. It is an evolutionary algorithm containing selection, recombination, and mutation. Because in general real valued functions are optimized, the recombination takes the mean of a number of parents and the mutation adds normal distributed random numbers to the variables. μ and λ denote the number of parents and offspring during the optimization.

Every individual represents a sample consisting of a translational and a rotational component like in Section 5.5.2. Depending on the type of evolutionary algorithm it may contain additional information which serves for the control of the optimization process.

The MO-CMA-ES has been applied with initial populations of size 25 from regular sampling and from LHS, choosing the best 25 individuals. 50 iterations have been performed. Figures 5.4 and 5.5 for the small workpiece and Fig. 5.7 and 5.8 for the large workpiece show that a visible improvement of the Pareto front of regular sampling and LHS is achieved. Moreover, the results with regular sampling and LHS as initialization are similar, although the number of feasible LHS samples is rather low. This indicates that the time-saving sparse sampling is sufficient for initializing the evolutionary algorithm.

The single-objective CMA-ES has been applied with an initial standard deviation of 50 mm for the normal distributed mutation, starting from the best individual of LHS. The $(\mu + \lambda)$-ES has been applied for $\mu = 25$ and $\lambda = 50$ and iterated over 50 generations, also with the initial population selected as the best individuals from regular sampling and from LHS. Figures 5.6 and 5.9 show the development of the objective function depending on the number of iterations.

The timings and fitness of the final result of all evolutionary variants are compiled in Tables 5.1 and 5.2. For the multi-objective variants, the resulting fitness is calculated with the same weights of the sub-objectives as for the single-objective variants. It can be seen that the CMA-ES and the $(\mu + \lambda)$-ES reach the best results, especially for the smaller workpiece. Due to parallelization, the $(\mu + \lambda)$-ES has the lower runtime despite using more function evaluations.

5.7 Local Descent

Local optimization methods [84] generate a sequence of samples by locally analyzing the objective function at the current sample in order to determine a direction into which the next one is chosen. A typical approach is to take a direction in which the best local improvement of the objective function value may be achieved. If the objective function is continuous of sufficient order, derivative information can be employed to choose the direction. For example, the *gradient* of the objective function leads to the direction of highest steepness. Starting from this observation, a number of modifications, like e.g. the *conjugate gradient method* or the *BFGS method* [84], have been developed.

For non-smooth objective functions, a local or global *meta-model*, also called *response surface*, can be used instead of the objective function or to calculate a descent direction. An example for a meta-model is a quadratic function which is generated using the *Box-Behnken design* [19].

A different approach for non-smooth objective functions are *derivative-free optimization methods*. Examples are the *Nelder-Mead algorithm* and *Pattern Search* [84]. Derivative-free methods can be applied if the gradient calculation is not feasible or if it is too time consuming to calculate.

The investigations of this chapter include the gradient descent approach in two versions. The first version calculates the gradient of the objective function by finite differences. The second version employs the Box-Behnken response surface method to calculate the descent direction and step length. The Box-Behnken design has been initially scaled to ± 20 mm and is then adapted to half of the previous step length.

As an example of a derivative-free approach, the Nelder-Mead algorithm has been chosen. The Nelder-Mead algorithm is initialized by creating a simplex from the initial position by translating it in each coordinate direction by 50 mm.

The results in Table 5.1 and Fig. 5.6 for the small workpiece show that the Nelder-Mead algorithm can improve the solution from regular sampling notably, while the gradient descent methods get stuck in a local optimum. The results for the large workpiece can be found in Fig. 5.9 and Table 5.2. The difference of the fitness achieved is less than for the small workpiece. This can be attributed to the constraints which restrict the optimization more than for the small workpiece. Also, note that due to different capabilities for parallelization the number of objective function evaluations is not directly related to the runtime. For example, the Nelder-Mead algorithm is not well-suited for parallelization.

5.8 Relaxation of the Workpiece Position

This section is concerned with the first method of relaxation, which has been motivated in Section 5.1. The idea of relaxation of the workpiece position is to allow a varying workpiece position along the manipulator path during the optimization, but imposing a soft constraint as a sub-objective demanding that the variation is minor. From the resulting workpiece path a workpiece position is derived, e.g. as the mean position over the path. This position is checked if it solves the original placement problem (5.33) for X = P.

5.8.1 Problem Specification

In the following, the approach of relaxation is considered for the kinematics case. It is formulated as a path optimization problem according to (5.7), but simplified analogously to (5.33) as

$$\min_{\hat{\mathbf{x}}} F(\hat{\mathbf{x}}) \tag{5.35}$$

subject to

$$\mathbf{h}_\mathrm{s}(\hat{\mathbf{x}}) \geq \mathbf{0}, \tag{5.36}$$

$$\mathbf{h}_\mathrm{a}(\hat{\mathbf{x}}) = \mathbf{0}, \tag{5.37}$$

$$\mathbf{h}_\mathrm{c}(\hat{\mathbf{x}}) > \mathbf{0}. \tag{5.38}$$

To simplify the notation, $\mathbf{x}$ is used instead of $\hat{\mathbf{x}}$ from here on. The kinematic chain of the manipulator is in some sense extended by an additional transformation which describes the workpiece position relative to the end-effector. This transformation is given by the tool path, and the resulting structure now has the workpiece position as “end-effector”. The path of the workpiece is mathematically given by

$${}^\mathrm{W}\mathbf{T}_\mathrm{P}(t) = {}^\mathrm{W}\mathbf{T}_\mathrm{M}{}^\mathrm{M}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi}(t))^\mathrm{T}\mathbf{T}_\mathrm{P}(t),\ t \in [0, t_f]. \tag{5.39}$$

The workpiece path ${}^\mathrm{W}\mathbf{T}_\mathrm{P}(t)$ can alternatively be written using an explicit representation as described in Section 5.3.1 as

$$\mathbf{x}_\mathrm{P}(t) = (\boldsymbol{\varphi}_\mathrm{P}(t), \mathbf{o}_\mathrm{P}(t))^\top,$$

with the rotational part $\boldsymbol{\varphi}_\mathrm{P}(t)$ being a quaternion in this case.

The objective is that the workpiece does not move. This can be expressed mathematically by additional sub-objectives limiting the motion of the workpiece path which are defined according to Section 5.3.1, i.e.

$$L_\mathrm{a,wr} = \| \log(\boldsymbol{\varphi}_\mathrm{P}^{-1}(t) \cdot \overline{\boldsymbol{\varphi}}_\mathrm{P})\|^2, \tag{5.40}$$

$$L_\mathrm{a,wt} = \|\mathbf{o}_\mathrm{P}(t) - \overline{\mathbf{o}}_\mathrm{P}\|_2^2, \tag{5.41}$$

where $\|\cdot\|$ in $L_{\mathrm{a,wr}}$ is the quaternion norm, $\log(\cdot)$ is the quaternion logarithm, $\boldsymbol{\varphi}_{\mathrm{P}}^{-1}(t)$ is the quaternion inverse (cf. Kuffner [65], Section V), and with the mean rotation $\overline{\boldsymbol{\varphi}}_{\mathrm{P}}$ and the mean translation

$$\overline{\mathbf{o}}_{\mathrm{P}} = \frac{1}{t_f}\int_0^{t_f} \mathbf{o}_{\mathrm{P}}(t)\,\mathrm{d}t. \tag{5.42}$$

Intuitively, the norm of the logarithm of a quaternion returns half of its rotation angle, while the product $\boldsymbol{\varphi}_{\mathrm{P}}^{-1}(t)\cdot\overline{\boldsymbol{\varphi}}_{\mathrm{P}}$ calculates the minimum rotation between the two orientations.

The mean rotation is calculated in the discrete setting from a sequence of quaternions by summing them up followed by a normalization. During the incremental summation, the next quaternion has to be near the partial sum, i.e. the Euclidean inner product of the two quaternions has to be non-negative, which is realized by selecting the suitable one from the two redundant quaternions $\boldsymbol{\varphi}_{\mathrm{P}}$ and $-\boldsymbol{\varphi}_{\mathrm{P}}$, i.e.

$$\overline{\boldsymbol{\varphi}}_{\mathrm{P}} = \frac{\sum_{j=0}^{m}\hat{\boldsymbol{\varphi}}_{\mathrm{P}}(t_j)}{\|\sum_{j=0}^{m}\hat{\boldsymbol{\varphi}}_{\mathrm{P}}(t_j)\|}\,, \tag{5.43}$$

for the correct selection $\hat{\boldsymbol{\varphi}}_{\mathrm{P}}(t_j)$, $j=0,\ldots,m$. In the continuous setting the quaternions would have to be integrated on the unit sphere in 4D.

The approach of solution presented here refers to a slightly reduced version of the objective function (5.16) which additionally includes the two new sub-objectives:

$$L = \lambda_\alpha\cdot L_\alpha + \lambda_{\mathrm{c}}\cdot L_{\mathrm{c}} + \lambda_{\mathrm{col}}\cdot L_{\mathrm{col}} + \lambda_{\mathrm{l}}\cdot L_{\mathrm{l}} + \lambda_{\mathrm{a,wr}}\cdot L_{\mathrm{a,wr}} + \lambda_{\mathrm{a,wt}}\cdot L_{\mathrm{a,wt}}\,. \tag{5.44}$$

The reduction concerns the energy which is not relevant for kinematics, and one of the manipulability measures. The only hard constraints taken into account are the joint angle limits from (5.12).

5.8.2 Solution

The solution uses the B-spline curve concept as a basis function representation analogously to (5.25) for the state function,

$$\mathbf{x}(\mathbf{x}_0,\ldots,\mathbf{x}_n,t) = \sum_{i=0}^{n}\mathbf{x}_i\cdot M_i(t). \tag{5.45}$$

Putting this and its required derivatives in the objective function, which has a form analogously to (5.15), yields a multidimensional real function in $\mathbf{x}_i$, $i=0,\ldots,n$,

$$F(\mathbf{x}_0,\ldots,\mathbf{x}_n) = \int_0^{t_f} L(\mathbf{x}(\mathbf{x}_0,\ldots,\mathbf{x}_n,t),\dot{\mathbf{x}}(\mathbf{x}_0,\ldots,\mathbf{x}_n,t),\ddot{\mathbf{x}}(\mathbf{x}_0,\ldots,\mathbf{x}_n,t))\,\mathrm{d}t. \tag{5.46}$$

Due to the convex-hull property of B-spline curves, the application of the joint angle constraints to the respective components of the control points $\mathbf{x}_i$ is sufficient to satisfy the constraints on the whole curve. The convex-hull property means that a curve is a subset of the convex hull of its control points. This way the problem has been reduced to an optimization problem over a multidimensional real space with box constraint. Due to this property, the optimization can be performed using the L-BFGS-B method [84]. The BFGS method is a quasi-Newton method which uses the gradients of the combined objective function and the steps during the optimization to approximate the inverse Hessian matrix. The limited-memory BFGS method uses only the last few steps to calculate this approximation, thus saving memory by not storing the inverse Hessian matrix approximation. The L-BFGS is also extended to handle box constraints on the variables, resulting in the L-BFGS-B method. For line search a simple inexact bisection line search using the Wolfe conditions is applied.

The initial solution of the relaxed optimization should be a rather simple manipulator path which induces a workpiece path with a minimal amount of collisions. For that purpose, a constant, non-moving manipulator path is applied. The constant manipulator configuration of the path should minimize the collisions and promises a good optimization result. The aim is achieved by sampling the configuration space of the last three manipulator joints regularly, or by taking a user-supplied initial configuration.

Alternative strategies for a good initial solution can of course choose varying manipulator configurations along the path. A valid manipulator path with respect to collision and reachability could be derived from a fixed workpiece position, by choosing arbitrary configurations for the unreachable tool poses.

The calculation of the gradients can be performed by finite differences. By applying the chain rule, the calculation of the partial derivatives for the control points can be separated from the calculation of the derivative of the objective kernel L,

$$\begin{aligned}\frac{\partial F(\mathbf{x}_0,\dots,\mathbf{x}_n)}{\partial \mathbf{x}_j} = \int_0^{t_f} & \frac{\partial L(\mathbf{x}(.,t),\dot{\mathbf{x}}(.,t),\ddot{\mathbf{x}}(.,t))}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}(\mathbf{x}_0,\dots,\mathbf{x}_n,t)}{\partial \mathbf{x}_j} \\ &+ \frac{\partial L(\mathbf{x}(.,t),\dot{\mathbf{x}}(.,t),\ddot{\mathbf{x}}(.,t))}{\partial \dot{\mathbf{x}}} \cdot \frac{\partial \dot{\mathbf{x}}(\mathbf{x}_0,\dots,\mathbf{x}_n,t)}{\partial \mathbf{x}_j} \\ &+ \frac{\partial L(\mathbf{x}(.,t),\dot{\mathbf{x}}(.,t),\ddot{\mathbf{x}}(.,t))}{\partial \ddot{\mathbf{x}}} \cdot \frac{\partial \ddot{\mathbf{x}}(\mathbf{x}_0,\dots,\mathbf{x}_n,t)}{\partial \mathbf{x}_j} \mathrm{d}t.\end{aligned}$$

This separation leads to a lower runtime in the gradient calculation, because the partial derivatives of the objective kernel L, which are the most time consuming part, do not directly depend on the control point variables and thus can be reused for all $\mathbf{x}_j$.

The integrals may be calculated approximately by replacing them by Riemann sums like in Eq. (5.28).

Because the stability of the workpiece is only a soft constraint, it cannot be expected that the result fulfills the constraint perfectly. Thus, in a post-processing step, the resulting tool path motion is first stabilized by taking the mean according to (5.43) and (5.42). If the result is feasible, it is taken as solution. Otherwise, a post-processing step according to Section 5.9.2 is applied.

5.8.3 Evaluation

The small workpiece has been used for evaluation. The tool path was given by 3714 data points. The manipulator path was represented as a multidimensional B-spline curve with 371 control points. The integral was approximated as a Riemann sum with 1238 data points, except for the term L_{col} for which only 371 data points have been used for reasons of computational efficiency.

The initialization of the iteration was performed by regular configuration sampling, but restricted to the last three joints of the ABB IRB 4600, for efficiency reasons. The objective function values from initialization are shown in Fig. 5.10. The initial solution used is $(0, 0, 0, 0, -\pi/4, 0)$, which is also marked in Fig. 5.10.

The weights of the combined objective function (5.44) were $(0.1, 10^{-3}, 0.1, 1, 10^6, 0.01)$. The objective function values of the combined objective and of the sub-objectives are

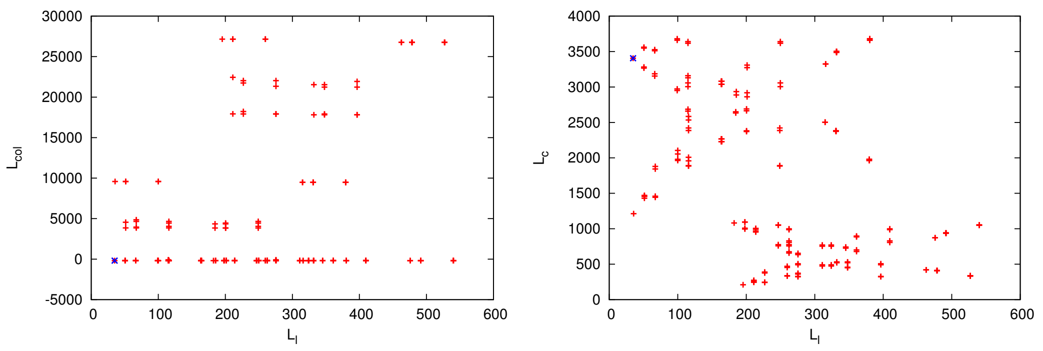


Fig. 5.10: Relaxation of the workpiece position of Section 5.8: Objective function values found by sampling the configuration space for the smaller workpiece. The marked element is used as initial solution.

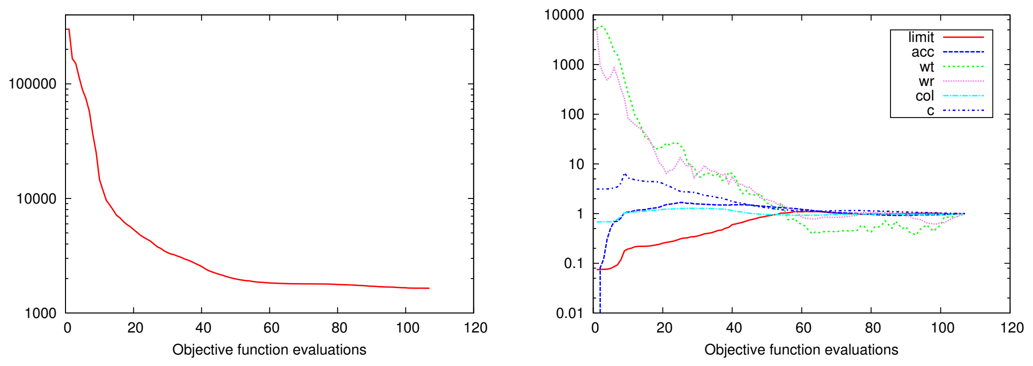


Fig. 5.11: Relaxation of the workpiece position of Section 5.8: Objective function values during optimization relative to the number of function evaluations for the smaller workpiece for the combined objective (left) and the sub-objectives (right). The objective values are scaled and the collision distance objective is additionally translated to fit into a single diagram.

shown in Fig. 5.11. The result achieves a minimization of the workpiece path variation, a collision-free path and in addition an optimization of the acceleration and manipulability. Post-processing was not necessary in this case. In Fig. 5.11 the progression of the objective values show that the workpiece movement is reduced while the other objectives increase their value. Especially the acceleration, which is zero at the start (not visible in the log-diagram), increases during the optimization.

The runtime of the initialization was 145 s (225 evaluations), and the runtime of the optimization was 845 s (138 evaluations of functions and gradients). Thus, the runtime is considerably higher than those of the separated methods. However, for manipulators with redundant axes it can be expected that the difference reduces since in that case the sub-problem of path optimization of the separated methods are of the same type as the relaxed problem.

5.9 Relaxation of Tool Path Following

The idea of relaxation of tool path following is to give up the constraint (5.13) that the manipulator has to exactly follow the tool path. This means that the objective (5.26) now becomes a penalty term in the objective function. In Section 5.4 it has replaced the constraint (5.13) for technical reasons of the representation, but it still acted as a hard constraint. Employing the norm described in Section 5.3.1 in fact again leads to two different objectives for translational and rotational distance.

The method of relaxation of tool path following may also simultaneously optimize the manipulator path and the relative placement of the workpiece and the manipulator, as the method of the preceding section can. This case is treated in Section 5.9.1. However, it can also be employed in variants were the placement remains unchanged and just the manipulator path is optimized. One such variant is presented in Section 5.9.2 where an optimized manipulator path has to be found whose end-effector path should have just minor deviations from the tool path. Another variant is described in Section 5.9.3 where the goal is an optimized manipulator path whose end-effector moves a spray gun so that the coating thickness is optimized as well. Both variants may be applied for post-processing of the placement methods of Sections 5.8 and 5.9.1.

5.9.1 Combined Manipulator Path and Placement Optimization

This approach may be seen as an alternative to the relaxation of the workpiece position for workpiece placement. It results from extending the optimization over the workpiece position ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{P}}$ of the separated approach with the relaxation of the tool path following.

Additionally, redundancies, e.g. a symmetric tool impact (symmetric spray cone), can be incorporated into the distance metric by taking the distance to a direction instead of the distance a whole orientation.

A practical issue when optimizing such different types of variables (manipulator path, workpiece position) is a potentially large difference in the magnitude of the derivatives, which could hinder the optimization process. This is remedied by variable scaling, i.e. the variables are multiplied by a scaling factor which is introduced into the objective functions. This also changes the scaling of the derivatives. The resulting optimization problem is solved with the method of Section 5.8.2.

The approach has been experimentally evaluated on basically the same configuration as in Section 5.8.3. The workpiece position is initialized as the world reference frame, while the rest of the initialization is the same as in Section 5.8.3. The only change of the weights concerns $\lambda_{\mathrm{a,wr}}$, which is set to 1000. The variable scaling mentioned above is 80 for the translational part and 0.5 for the rotational part of the workpiece pose. The progression of the objective function is displayed in Fig. 5.12(a-b). The progressions of the individual objectives show that mostly the very large distance to the workpiece and the predefined tool path are minimized, while the acceleration along the manipulator path increases during the optimization. Most of the other objectives change only by a smaller factor, except for the collision avoidance objective, which also has a high value at the beginning of the optimization. The runtime of the optimization was 940 s.

The coating error, which is not part of the optimization, increases in comparison to the optimized tool path (cf. Fig. 5.13(a-b)). This is due to the deviation of the end-effector path from the given tool path. The maximum and root mean square error are again defined relative to the target coating height, as in Section 2.8. The tool path of Fig. 5.13(a) is the same that is used throughout this chapter, optimized according to Chapter 2. As in

Chapter 2, the boundary is excluded from the deposition simulation to gain more meaningful error values. The reason for the deviation from the given tool path, which results in the larger coating error, is the simultaneous optimization of the manipulator path. The different objective values, e.g. path-following and acceleration minimization, are balanced against each other such that a smaller acceleration is achieved at the cost of a larger distance to the given tool path. The larger error values near the turning points of the path result from a change in the distance between consecutive path segments, due to the rounded turning path element.

The potential increase in the coating error, as witnessed in the previous example, makes it necessary to subsequently optimize the coating deposition, e.g. according to Section 5.9.3.

5.9.2 Separate Manipulator Path Optimization

The relaxation of the tool path following can also be performed with a static workpiece position. In that case, only the shape of the manipulator path is affected. This is especially useful as a post-processing step for the relaxation of the workpiece position, or as a basis for the reoptimization of the coating height in the next section.

Figure 5.12(c-d) shows the progression of the objective function of an experimental evaluation on the set-up employed before. The workpiece position was set to the optimized position from the optimization of the previous subsection, but the manipulator path was reset to a constant manipulator configuration. This results in a similar progression of the objective functions (cf. Fig. 5.12(a-b)), with the only difference that the initial distance to the predefined tool path is not as high. The runtime of the optimization was 910 s. The coating error (Fig. 5.13(c)) also increases similarly to the optimization of the previous subsection, which again makes a subsequent optimization of the coating deposition necessary, cf. Section 5.9.3. This post-processing step can be omitted if the distance to the given tool path is too small to significantly affect the coating error.

5.9.3 Manipulator Path Optimization with Coating Error Minimization

The deviation from the predefined tool path caused by the relaxation of the constraint of tool path following of the previous subsections may cause errors for the application which might not be acceptable. In this section, a method of error reduction for spray coating is presented which may help to reduce the error. The method allows to modify the given spray tool path and extends the objective function by a further objective which minimizes the coating error.

The new sub-objective is

$$F_{\text{coat}} = \frac{1}{A_M} \int_M \left(h^s(\mathbf{p}) - \int_0^{t_f} h({}^{\mathrm{W}}\mathbf{T}_{\mathrm{P}}(\mathbf{p}), {}^{\mathrm{W}}\mathbf{T}_{\mathrm{M}} {}^{\mathrm{M}}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi}(t)))\, \mathrm{d}t \right)^2 \mathrm{d}M(\mathbf{p}). \tag{5.47}$$

${}^{\mathrm{W}}\mathbf{T}_{\mathrm{M}} {}^{\mathrm{M}}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi}(t))$ is the end effector path used as tool path. The first term of the integrand represents the desired coating height, and the second term the deposited material at a point $\mathbf{p}$ on the workpiece surface M (cf. Chapter 2).

Since spray time is an important parameter of spray path optimization, the domain of optimization is extended with respect to this sub-objective by the possibility of reparametrization with respect to time as well, in contrast to the location-based path optimization used up to now. For this purpose, an alternative time line s, $s \in [0, s_f]$, is introduced over which the new objective (5.47) is considered:

$$F_{\text{coat}} = \frac{1}{A_M} \int_M \left(h^s(\mathbf{p}) - \int_0^{s_f} h({}^{\mathrm{W}}\mathbf{T}_{\mathrm{P}}(\mathbf{p}), {}^{\mathrm{W}}\mathbf{T}_{\mathrm{M}} {}^{\mathrm{M}}\mathbf{T}_{\mathrm{E_M}}(\tilde{\boldsymbol{\varphi}}(s)))\, \mathrm{d}s \right)^2 \mathrm{d}M(\mathbf{p}). \tag{5.48}$$

The relation to the timeline t is established by a strictly monotonous time line function $s(t)$, $t \in [0, t_f]$. With this function, $\tilde{\boldsymbol{\varphi}}(s(t)) = \boldsymbol{\varphi}(t)$. This representation converts Eq. (5.48) into

$$F_{\text{coat}} = \frac{1}{A_M} \int_M \left(h^s(\mathbf{p}) - \int_0^{t_f} h({}^{\mathrm{W}}\mathbf{T}_{\mathrm{P}}(\mathbf{p}), {}^{\mathrm{W}}\mathbf{T}_{\mathrm{M}}{}^{\mathrm{M}}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi}(t))) \cdot \dot{s}(t)\, \mathrm{d}t \right)^2 \mathrm{d}M(\mathbf{p}), \tag{5.49}$$

where $\dot{s}(t)$ is the derivation of $s(t)$ for t. The positive function $\dot{s}(t)$ is the extension of the domain of optimization which represents the possibility of reparametrization. It extends the state vector (5.32) by a further component, i.e.

$$\tilde{\mathbf{x}}^\top = (\boldsymbol{\varphi}^\top, \dot{s}).$$

The basis function representation of Eq. (5.49) analogously to Eq. (5.46) is

$$\begin{aligned} F_{\text{coat}} = \frac{1}{A_M} \int_M \Bigg(h^s(\mathbf{p}) - \int_0^{t_f} & h({}^{\mathrm{W}}\mathbf{T}_{\mathrm{P}}(\mathbf{p}), {}^{\mathrm{W}}\mathbf{T}_{\mathrm{M}}{}^{\mathrm{M}}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi}(\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_n, t))) \\ & \cdot \dot{s}(s_0, \ldots, s_n, t)\, \mathrm{d}t \Bigg)^2 \mathrm{d}M(\mathbf{p}), \end{aligned} \tag{5.50}$$

where $s_0, \ldots, s_n$ are the control parameters of a B-spline representation of $\dot{s}(t)$.

The corresponding discretized version of Eq. (5.49) which is used for the approximate solution is

$$\begin{aligned} F_{\text{coat}} \approx \frac{1}{A_M} \sum_{i=0}^{N} \Bigg(h^s(\mathbf{p}_i) - \sum_{j=0}^{m} & h({}^{\mathrm{W}}\mathbf{T}_{\mathrm{P}}(\mathbf{p}_i), {}^{\mathrm{W}}\mathbf{T}_{\mathrm{M}}{}^{\mathrm{M}}\mathbf{T}_{\mathrm{E_M}}(\boldsymbol{\varphi}(\boldsymbol{\varphi}_0, \ldots, \boldsymbol{\varphi}_n, t_j))) \\ & \cdot \dot{s}(s_0, \ldots, s_n, t_j) \cdot \Delta t_j \Bigg)^2 \cdot A_i \,. \end{aligned} \tag{5.51}$$

The discretization over the outer integral follows Section 2.5.3. The Δt_j, $j = 0, \ldots, m$, are those from Eq. (5.29).

The rest of the objective function remains mostly unchanged since almost all the sub-objectives concern the shape of the manipulator path. For those the reparametrization of the manipulator path is irrelevant. An exception is the sub-objective F_α. Furthermore, the motion-related constraints (5.12) are affected. In those cases the reparametrization has to be taken into account, which can be immediately achieved by differential calculus:

$$\tilde{\boldsymbol{\omega}}(s) = \frac{\mathrm{d}\tilde{\boldsymbol{\varphi}}(s)}{\mathrm{d}s} = \frac{\mathrm{d}\boldsymbol{\varphi}(t)}{\mathrm{d}t} / \dot{s}(t) = \boldsymbol{\omega}(t)/\dot{s}(t).$$

$$\tilde{\boldsymbol{\alpha}}(s) = \frac{\mathrm{d}^2\tilde{\boldsymbol{\varphi}}(s)}{\mathrm{d}s^2} = \frac{\mathrm{d}^2\boldsymbol{\varphi}(t)}{\mathrm{d}t^2} / \dot{s}^2(t) - \ddot{s}(t)/\dot{s}^2(t)\tilde{\boldsymbol{\omega}}(s) = \boldsymbol{\alpha}(t)/\dot{s}^2(t) - \boldsymbol{\omega}(t)\ddot{s}(t)/\dot{s}^3(t).$$

For an experimental analysis, the objective F_{coat} according to (5.51) is evaluated on 3714 samples along the manipulator path and has a weight of $\lambda_{\text{coat}} = 10^6$. The only difference to the method of Section 5.9.1 is that $\lambda_{\text{a,wt}}$ is decreased to 10^{-3} to allow more flexibility for the coating optimization. The workpiece position and manipulator path are initialized to the optimization result of the relaxation with a static workpiece. The progression of the objective function is shown in Fig. 5.12(e-f). Due to a very big coating error in the beginning, mostly the coating objective was minimized during the optimization. The translational distance to the predefined tool path and the acceleration of the manipulator increased to allow a minimization of the coating error. This means on one hand that the predefined tool path was not really suited for coating, but on the other hand it shows that the optimization seems to be able to compensate that. The overall runtime of the optimization was 1952 s.

The coating error is visualized in Fig. 5.14. In contrast to Fig. 5.13 the given tool path is not optimized and the coating height is considered on the whole surface, which results in a generally lower coating height and small peaks at the turning points of the path. This provides a greater challenge to the manipulator path optimization and thus is better suited to show the abilities of the optimization. After the placement optimization according to Section 5.9.1, the coating error slightly increases even more. The manipulator path optimization presented in this section reduces the coating error significantly (cf. Fig. 5.14(c), note the different scaling). In comparison to the optimized tool path of Fig. 5.13(a) the coating error is slightly larger, but the coating is optimized over the whole surface, which requires a rather large modification of the path geometry. Also, the additional objectives used in the manipulator path optimization again may increase the coating error due to the balancing resulting from the single weighted objective function.

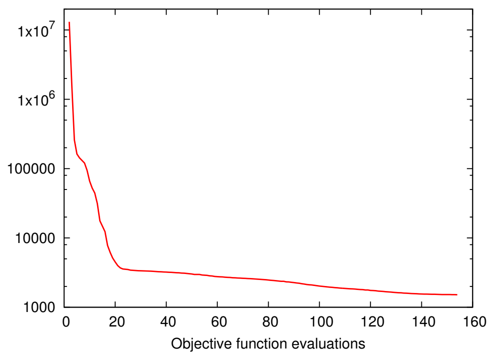
5.10 Discussion

According to Tables 5.1 and 5.2 the best results with respect to the fitness, i.e. the value of the objective function, have been achieved by the Nelder-Mead algorithm. Figures 5.15 and 5.16 show the results of the best placements for the small and the large workpiece. Due to the global nature of the preceding sampling very different placements can be achieved with the separated approach, which is not possible with the local optimization in the relaxed approaches, at least if only one initial solution is used. For comparison, the final placement of the relaxation methods for the small workpiece are displayed in Fig. 5.17 and 5.18. They are rather similar to each other, but differ from the result of the separated approach (Fig. 5.15) due to the difference between local and global optimization.

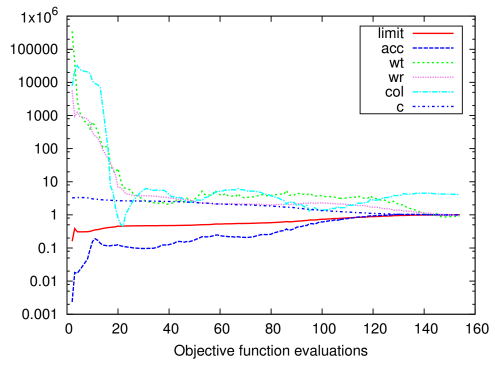
Apart from the global and local characteristics of the separated and relaxed approaches, respectively, the problems that are solved are different and thus the approaches are suited for different situations. The relaxed approaches, which are more complicated and do not return a global optimum, should only be applied if the separated approach is not feasible. This is the case if the tool path is not feasible or the feasible region for the tool path is too small, or if a manipulator path optimization is required, which would increase the runtime of the separated approach too much. A manipulator path optimization is required if the manipulator has redundant axes or the tool path is only partially defined.

One of the differences between the two relaxation approaches lies in the distance calculation. The relaxation of the workpiece position minimizes the motion of the workpiece by minimizing the distance to the mean workpiece position. The relaxation of the tool path following on the other hand minimizes the distance to the predefined tool path directly. The latter is more suited if the manipulator path deviates much from the predefined tool path to be feasible. Another difference between the two approaches is the dimension of the optimization problem. Both approaches have variables for the manipulator path, while only the relaxation of the tool path following needs additional variables for the workpiece position. It is also advisable to use the relaxation of tool path following if the tool path does not define all the degrees of freedom, e.g. for processes with symmetric tool impacts like milling or symmetric spray cones in spray coating.

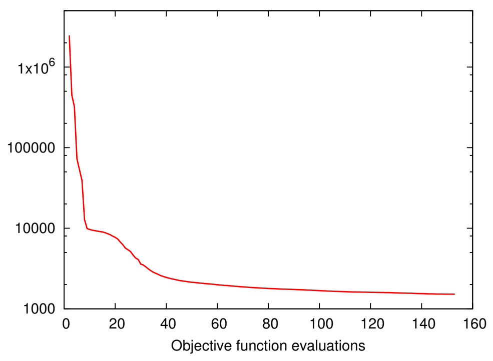
A further evaluation of the method from Section 5.9.1 was carried out for a more complex workpiece. It demonstrates the usefulness of the method for the case of a highly complex tool path, where suitable tool reference frames can only be guessed in advance, which makes



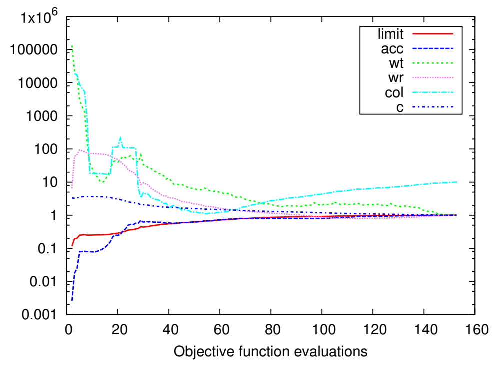
(a)



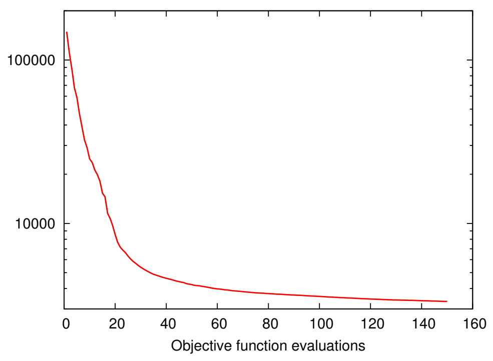
(b)



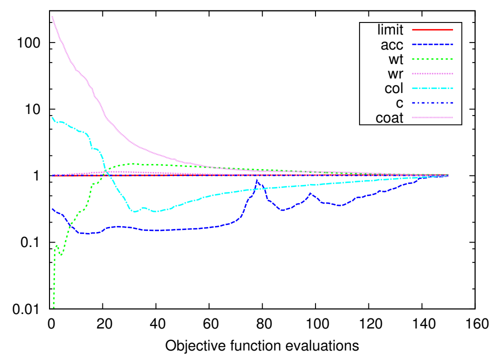
(c)



(d)

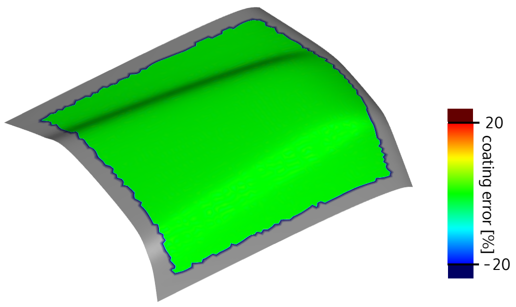


(e)

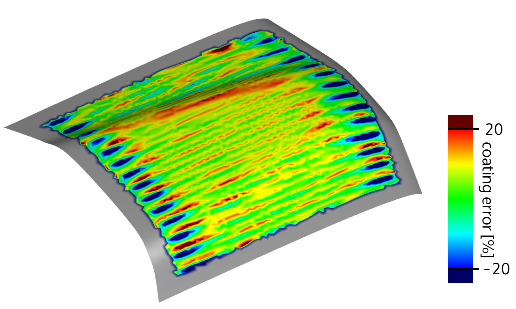


(f)

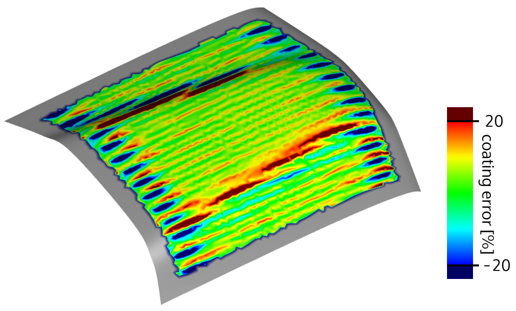
Fig. 5.12: Relaxation of tool path following of Section 5.9: Objective function values during optimization relative to the number of function evaluations for the smaller workpiece. The objective values are scaled and the collision distance objective is additionally translated by 170, 175, and 182, respectively, to fit into a single diagram. (a-b) method of Section 5.9.1, (c-d) method of Section 5.9.2, (e-f) method of Section 5.9.3.



(a) max error: 1.4%, rmse: 0.35%

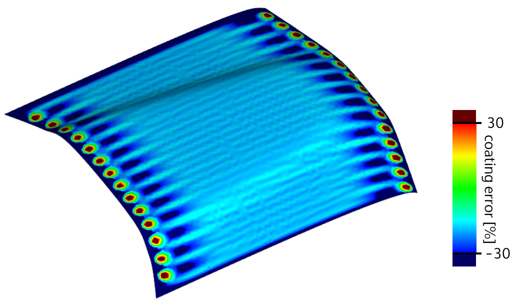


(b) max error: 47%, rmse: 11.9%

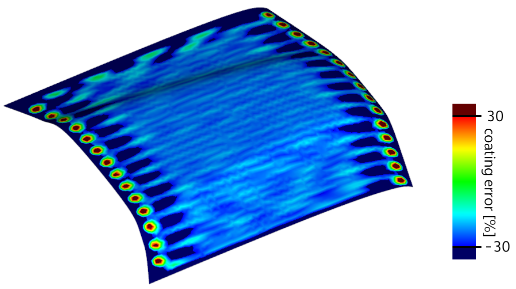


(c) max error: 82%, rmse: 14%

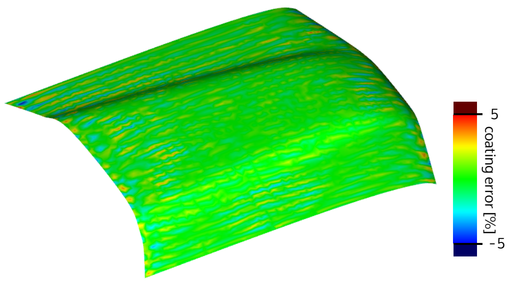
Fig. 5.13: Result of the application specific optimization w.r.t. the coating error (a) of the given tool path optimized according to Chapter 2, (b) after the placement optimization of Section 5.9.1, and (c) after the optimization of Section 5.9.2 for the smaller workpiece.



(a) max error: 100%, rmse: 31.3%



(b) max error: 100%, rmse: 34.3%



(c) max error: 5.6%, rmse: 1.21%

Fig. 5.14: Result of the application specific optimization w.r.t. the coating error (a) of the given tool path, (b) after the placement optimization of Section 5.9.1, and (c) after the coating optimization of Section 5.9.3 for the smaller workpiece. The boundary of the workpiece is excluded from the deposition simulation to have more meaningful error values.

a deviation from the given tool path necessary. Using the same initial solution as for the hood workpiece, the manipulator deviates from the predefined tool path (Fig. 5.19), because the tool path is not completely reachable in this case. Nevertheless, the minimization of the distance to the tool path produces a manipulator path which utilizes the feasible space. Rotating the sixth axis by 90 degrees in the initial solution results in a better approximation of the predefined tool path (Fig. 5.20). Another useful variation, especially for symmetric tool impacts, is to apply a rotational distance metric only to the spray direction, i.e. to measure only the deviation of the spray direction, which gives the manipulator more freedom for optimization (Fig. 5.21). The predefined tool path and the optimization results are shown in Fig. 5.22. It can be noticed that the optimized end-effector paths approximate the predefined tool path to a varying degree, depending on the initial solution and distance metric, i.e. if only the distance to the spray direction is used.

5.11 Concluding Remarks

Two algorithmic paradigms for path placement optimization have been considered, separated placement and path optimization and simultaneous placement and path optimization. The first approach is canonical for manipulators without redundant axes for which almost no degrees of freedom besides placement exist, but it can also be applied in presence of redundant axes. The computational performance of wide-spread optimization approaches has been investigated for separated path placement optimization, with the result that evolutionary approaches are best-suited, in particular on multi-core platforms.

For applications where reachability and collision are critical, the approach of separation may become ineffective since the subspace of feasibly solutions is small and many unsuccessful trials of placement are potentially executed. Moreover, the approach of separation is rather impractical for redundant manipulators due to a high runtime. For that reason, the simultaneous approach has also been applied. Two methods of relaxation have been proposed for this purpose. The optimization has been performed with a finite basis representation of the manipulator paths. Although the time consumption is notably higher than for the approach of separation, it is sufficient for the application purposes.

Future work may concern the inclusion of a more realistic manipulator model, especially concerning the manipulator dynamics. Another goal could be to include further constraints and objectives regarding the layout of the whole manipulator cell, where additional objects may exist. Performance improvements could be achieved by using a faster distance calculation for collision avoidance. A method to estimate the penetration depth [113] could be helpful to quantify the infeasibility during optimization. Other methods to quantify the collision include the intersection volume [110] or a criterion based on the distance function of the obstacles and an approximation of the manipulator by spheres [88].

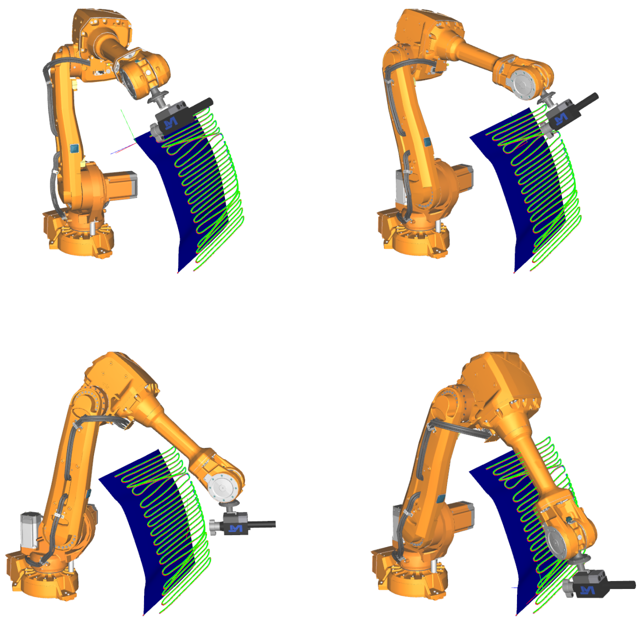


Fig. 5.15: Resulting placement of the Nelder-Mead algorithm of Section 5.7 for the smaller workpiece.

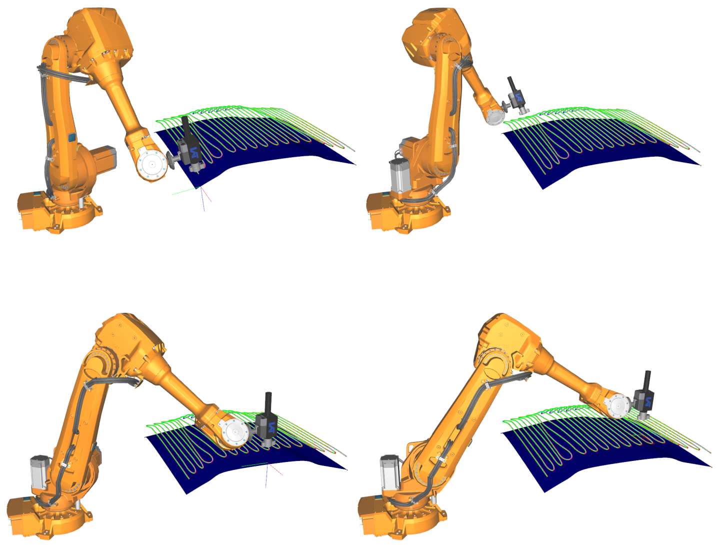


Fig. 5.16: Resulting placement of the Nelder-Mead algorithm of Section 5.7 for the larger workpiece.

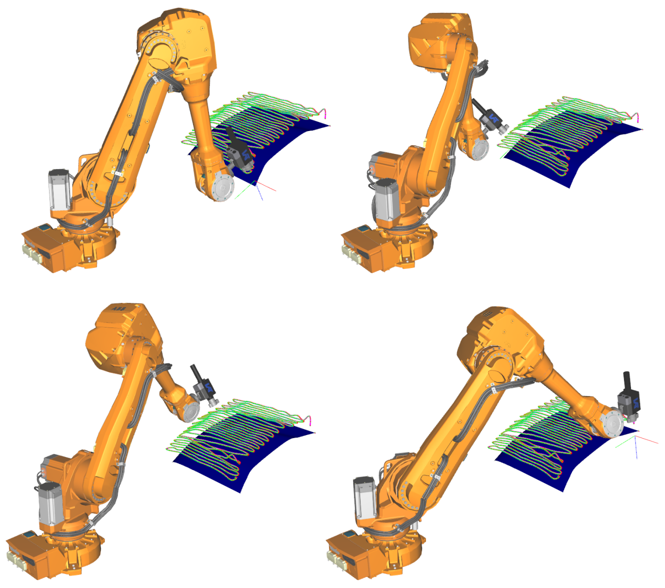


Fig. 5.17: Optimization result of the combined manipulator path and placement optimization of Section 5.8 for the smaller workpiece, before post-processing.

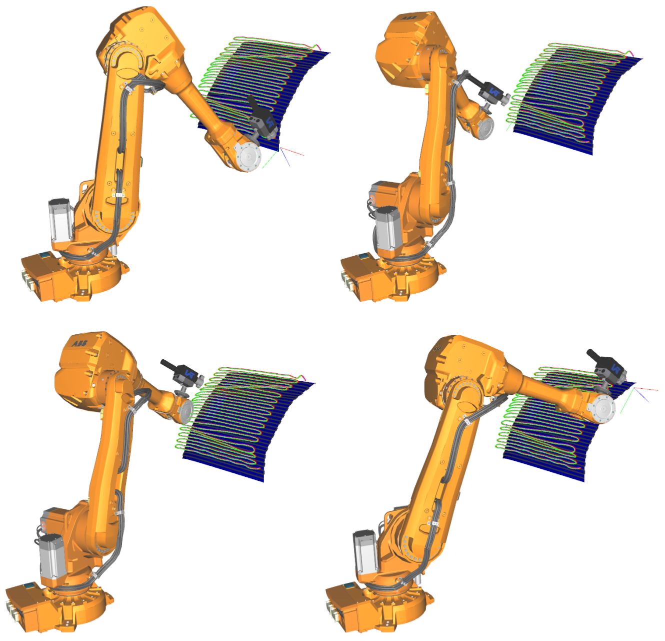


Fig. 5.18: Optimization result of the combined manipulator path and placement optimization of Section 5.9.1 for the smaller workpiece.

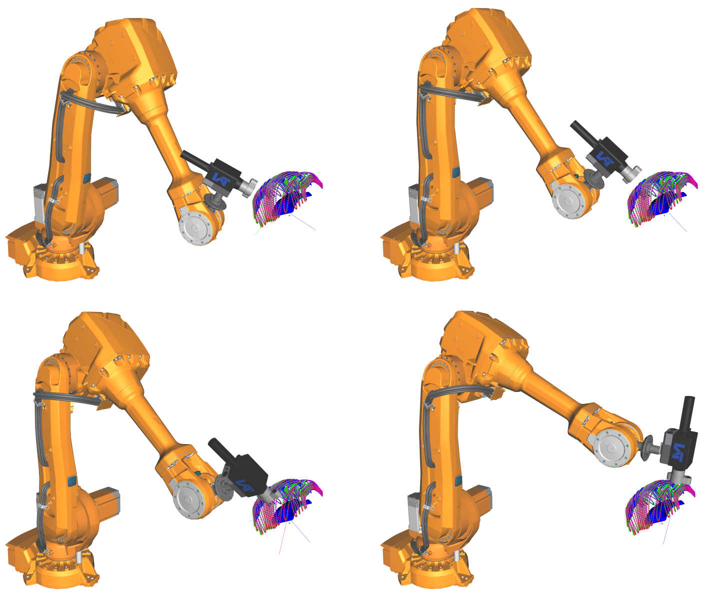


Fig. 5.19: Optimization result of the combined manipulator path and placement optimization of Section 5.9.1 for a more complex workpiece.

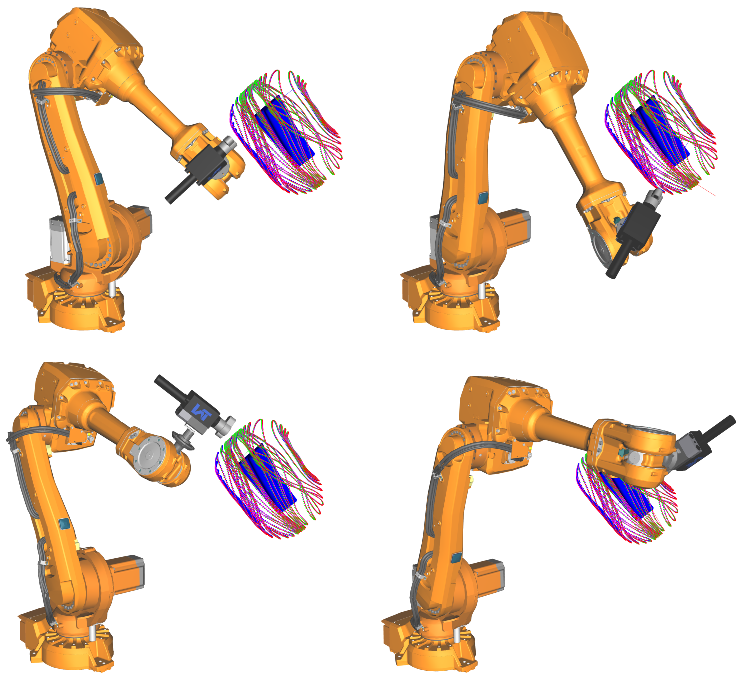


Fig. 5.20: Optimization result of the combined manipulator path and placement optimization of Section 5.9.1 for a more complex workpiece, with a different initial solution.

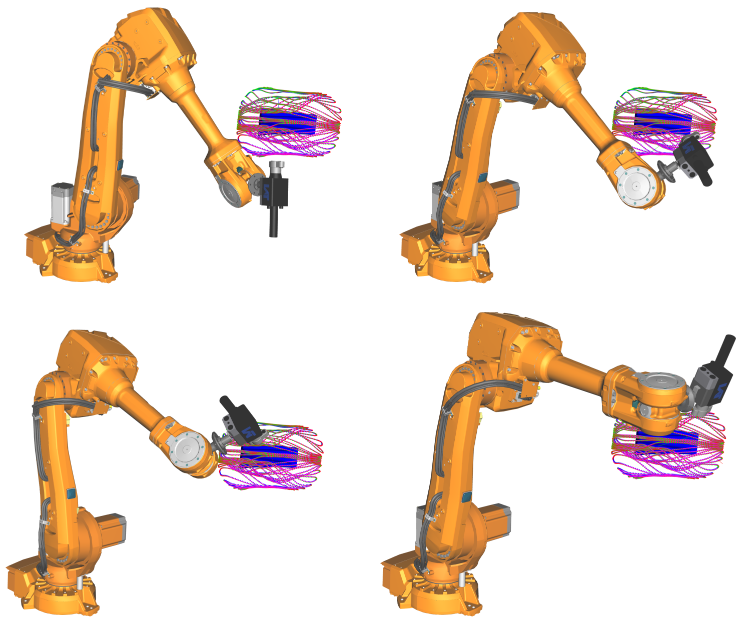


Fig. 5.21: Optimization result of the combined manipulator path and placement optimization of Section 5.9.1 for a more complex workpiece, with the rotational distance metric only considering the spray direction.

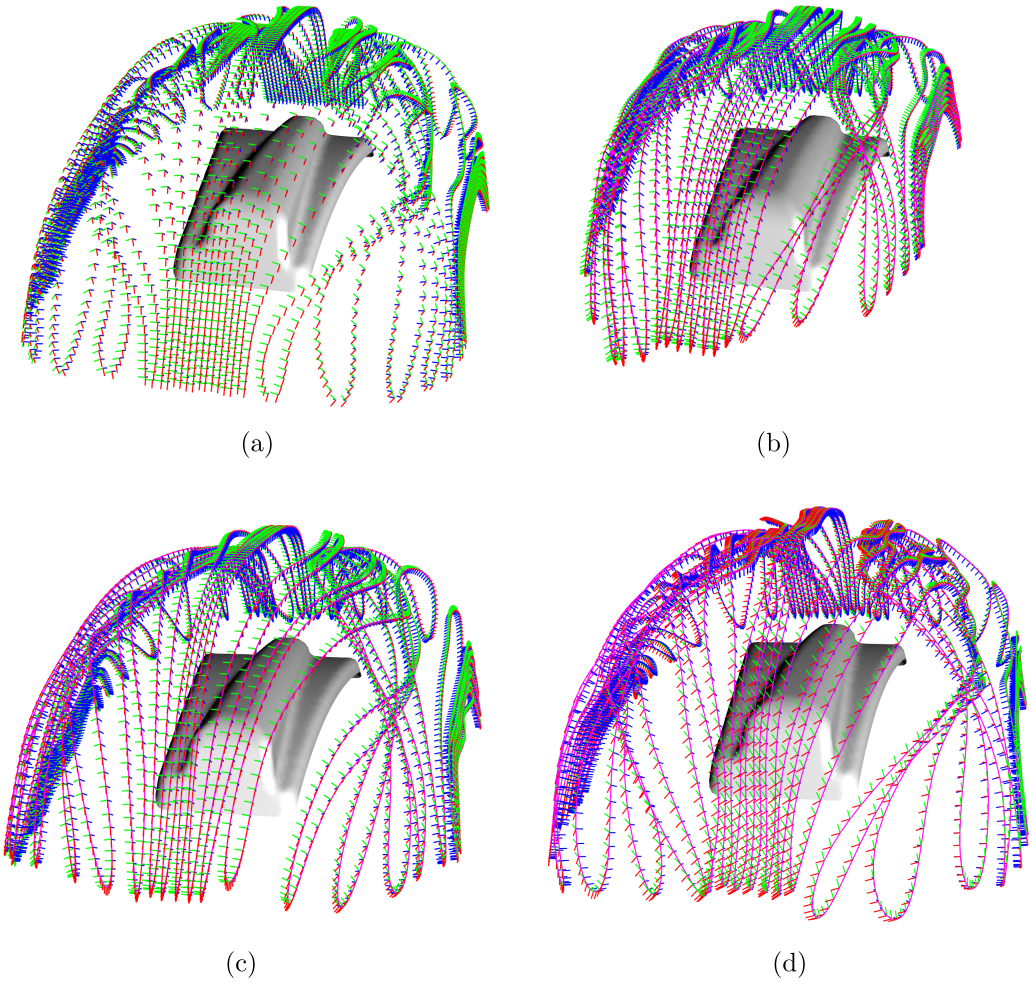


Fig. 5.22: Comparison of (a) the predefined tool path, (b) the optimized manipulator path from Fig. 5.19, (c) the optimized manipulator path from Fig. 5.20, and (d) the optimized manipulator path from Fig. 5.21.

6 Conclusion and Future Work

This chapter summarizes the results of the thesis and outlines directions of future work.

6.1 Summary

This thesis has made novel contributions to automatic path planning for path-oriented production techniques. They were mainly motivated by path planning for spray coating processes. In contrast to milling, those processes have found less interest in the basic research community in the past.

The starting point of the thesis, and its first result, was a novel two step approach for automatic path planning to achieve a desired spray coating height on free-form surfaces. It can be characterized as a path-geometry-last approach to tool path planning, in contrast to the widespread path-geometry-first approaches. The purpose of the first step is a path-free collection of process-related information in form of an optimized, so-called gun configuration cover. The major information is a spray time density, a spray gun density, and spray distance. This information is used in a second step for the construction of an appropriate gun path by resampling the optimized gun cover. The whole concept is based on impact path-based path planning. The second step was worked out for impact paths based on offset curves represented as isolines of surface distance functions.

This type of paths and curves has lead to the second emphasis of the thesis, improvements of the concept of offset curves. The advantage of those curves is that they are in widespread use and that the representation as isolines is stable and flexible. In this thesis they have been considered on surfaces represented by triangular meshes, taking into account the state-of-the-art of mesh processing. In the last two decades, the importance of triangular meshes in CAD and CAM has increased considerably, and a profound theoretic background has been developed in basic research. The thesis has addressed two thematic complexes.

The first thematic complex concerns the local adaptation of the offset distances to the requirements of production processes. For this purpose, a novel approach employing anisotropic distance functions on triangular meshes has been presented. The anisotropy has been achieved by metric tensors assigned to the mesh vertices from which the anisotropic distance function is calculated by a modification of the MMP algorithm. The resulting method of adaptation has been applied to derive milling paths with uniform cusp heights in a much more simple way than a previously known solution. A second application concerns the already mentioned second phase of the path-geometry-last approach and an error-adaptive path interval adaptation for spray coating.

The second thematic complex focused on diminishing of two major drawbacks of isolines, sharp corners and a complex topology. In contrast to simple qualitative approaches by low-pass filtering, a novel quantitative approach based on a multi-objective optimization problem has been presented. It is based on weighted distance functions on triangular meshes and employs a novel concept of weighted medial axes for regions with boundaries. The weighted distance functions are calculated by another modification of the MMP algorithm. The optimization is performed by a multi-objective evolutionary algorithm. An experimental evaluation has shown the usefulness of the approach.

The third emphasis of the thesis includes aspects of the manipulator into path planning. This has lead to a path placement problem where a given tool path has to be placed relatively to the manipulator so that it becomes executable in an optimized way. The problem has been formally specified as an optimization problem. An extensive experimental comparative analysis of different optimization methods for a six-axes industrial robot, i.e. a manipulator without redundant axes, and spray coating paths as use case has been performed. The methods included grid search, Latin hypercube sampling, evolutionary algorithms, Nelder-Mead search, and different gradient descent methods. It has turned out that evolutionary algorithms and the Nelder-Mead algorithm are best-suited.

Furthermore, different approaches of relaxation of the original problem have been presented, with the aim to increase the chance of finding a feasible solution in complex settings or an improved resulting solution. The first approach is to allow a time-dependent motion of the workpiece, with a penalty term for the amount of motion. The second approach allows a deviation from the given tool path, with a penalty term for the amount of deviation. The third approach is an extension of the second one which includes an application-dependent optimization of the tool path. The methods have been experimentally analyzed for a use case from spray coating. In the example the root mean square coating error has been reduced from about 33% to 1.2%.

The main advantage of the relaxed approaches is the suitability for redundant manipulators and for redundant tool paths, i.e. not fully predefined tool paths. Another crucial advantage is the simultaneous variation and optimization of the manipulator path, which allows to find a solution near the predefined tool path, even if there is no feasible solution for the tool path as it is. This is important, because it cannot be guaranteed in advance that a tool path has a feasible solution.

6.2 Future Work

The thesis has preferred offset paths as path type. It would be interesting to extend path planning for spray coating to other classes of paths. This aspect is of particular interest for thermal spray coating for which a further objective of path planning is to minimize the heat variance over the surface. In particular it is not necessary that the paths are free of self-intersections, as are the offset paths. Examples of such paths are the billiard paths by Duncan et al. [55], their generalization to boundary-to-boundary paths by Hegels et al. [46] and the trochoidal-like paths which have been applied by Elber et al. for high-speed milling [36]. An interesting question is whether the path-geometry-last approach can also be applied to the coating height optimization for those classes of paths in the second step.

Another related aspect is to employ manipulators with redundant axes with the aim to achieve an increased relative speed between the tool and the workpiece in order to avoid extreme local heating of the workpiece and the coating.

An extension of the path placement problem of practical interest is to consider constraints of the production process and the production cell. In particular for thermal spraying, spraying in direction of the manipulator should be avoided in order to prevent a self-destruction of the manipulator. A more complex problem with respect to the placement is to consider a complete robot cell with multiple objects and maybe even multiple workpieces. This problem combines aspects of the facility layout problem and scheduling with manipulator path planning and collision avoidance and is therefore rather challenging.

Most of the methods presented in the thesis, except for the relaxation approaches of Chapter 5, are discrete and approximative. This observation concerns in particular the resulting paths. In practice, the input is often a smooth CAD model, and the resulting paths should also be smooth, or at least at a resolution within the tolerance of the production system. Methods to bridge the gap are required. A possible approach is to interpolate and approximate the discrete paths and perform a post-optimization over the domain of the interpolation or approximation scheme relatively to the input workpiece. Post-optimization can probably be performed efficiently and effectively by a local iterative method since the discrete solution should already provide a good initial solution.

A further challenge is to incorporate dynamics, i.e. force-related aspects, of the manipulator into path planning. To this end, the dynamics model of Chapter 5 can be implemented for a specific manipulator and used in the optimization. Another possibility is to pursue a simulation-based approach of optimization where the simulation uses a more complicated model of the dynamics of the manipulator. A manipulator path obtained by one of the methods of the thesis may serve as a reasonable initial solution of such an iterative approach of optimization.

A Discrete Gradient

The discrete gradient of a scalar function φ on a triangular surface mesh can be calculated for each face of the mesh, where it is constant [79]. A different approach, which generalizes to vertices, views the gradient calculation as basis transformation. In the parameter space of each triangle the gradient is given by a two-dimensional vector containing the differences of vertex values along both edges defining the parameter space. This vector is transformed into $\mathbb{R}^3$ by solving the system of linear equations

$$\begin{pmatrix} (\mathbf{v}_1 - \mathbf{v}_0)^\top \\ (\mathbf{v}_2 - \mathbf{v}_0)^\top \\ \mathbf{n}^\top \end{pmatrix} \cdot \nabla\varphi = \begin{pmatrix} \varphi_1 - \varphi_0 \\ \varphi_2 - \varphi_0 \\ 0 \end{pmatrix},$$

where the $\mathbf{v}_i$ are the positions of the vertex v_i in $\mathbb{R}^3$, $\mathbf{n}$ is the normal vector of the triangle, the φ_i are the vertex values and $\nabla\varphi$ is the resulting gradient vector. If only the two-dimensional gradient in the tangent space of the triangle is to be calculated, then the last row containing the normal vector has to be omitted. This system of linear equations can be solved by multiplying it with the inverse matrix

$$\frac{1}{2A_T} \begin{pmatrix} ((\mathbf{v}_0 - \mathbf{v}_2)^\perp)^\top \\ ((\mathbf{v}_1 - \mathbf{v}_0)^\perp)^\top \\ 2A_T\mathbf{n}^\top \end{pmatrix}^\top,$$

which contains column vectors orthogonal to the vectors of the original matrix, with $\perp$ indicating a 90° rotation and A_T being the triangle area. The inverse can be multiplied from the left because of the property $AA^{-1} = A^{-1}A$, i.e.

$$\frac{1}{2A_T} \begin{pmatrix} ((\mathbf{v}_0 - \mathbf{v}_2)^\perp)^\top \\ ((\mathbf{v}_1 - \mathbf{v}_0)^\perp)^\top \\ 2A_T\mathbf{n}^\top \end{pmatrix}^\top \cdot \begin{pmatrix} (\mathbf{v}_1 - \mathbf{v}_0)^\top \\ (\mathbf{v}_2 - \mathbf{v}_0)^\top \\ \mathbf{n}^\top \end{pmatrix} \cdot \nabla\varphi =$$

$$\frac{1}{2A_T} \begin{pmatrix} ((\mathbf{v}_0 - \mathbf{v}_2)^\perp)^\top \\ ((\mathbf{v}_1 - \mathbf{v}_0)^\perp)^\top \\ 2A_T\mathbf{n}^\top \end{pmatrix}^\top \cdot \begin{pmatrix} \varphi_1 - \varphi_0 \\ \varphi_2 - \varphi_0 \\ 0 \end{pmatrix}.$$

The gradient is therefore

$$\nabla\varphi = (\varphi_1 - \varphi_0) \cdot \frac{(\mathbf{v}_0 - \mathbf{v}_2)^\perp}{2A_T} + (\varphi_2 - \varphi_0) \cdot \frac{(\mathbf{v}_1 - \mathbf{v}_0)^\perp}{2A_T}. \tag{A.1}$$

To extent this approach to vertices, the edges incident to a vertex first are projected or rotated into the tangent plane of the vertex, which results in tangent vectors $\mathbf{t}_1, \ldots, \mathbf{t}_n$. The tangent plane is defined by the vertex normal which may be estimated on meshes by the discrete Laplace-Beltrami operator [79]

$$\Delta_M \mathbf{v}_i = \frac{1}{2A_i} \sum_{j \epsilon N_1(i)} (\cot\alpha_{ij} + \cot\beta_{ij})(\mathbf{v}_j - \mathbf{v}_i) \tag{A.2}$$

where A_i is the area of the Voronoi region around vertex v_i, α_{ij} and β_{ij} are the two angles opposite to the edge from vertex v_i to v_j, $\mathbf{v}_i$ and $\mathbf{v}_j$ are the coordinates of v_i and v_j, and $N_1(i)$ is the 1-ring neighborhood of vertex v_i (cf. Fig. 2.2). The estimation is motivated by continuous differential geometry where

$$2H\mathbf{n} = \Delta_M \mathbf{v}\,, \tag{A.3}$$

where $\mathbf{n}$ is the normalized normal vector, H is the mean curvature of the surface, and $\Delta_M \mathbf{v}$ is the Laplace operator at $\mathbf{x}$ which is approximated at mesh vertices by the discrete Laplace-Beltrami operator (Eq. A.2).

Then a similar system of linear equations is constructed for the edges incident to the vertex:

$$\begin{pmatrix} \mathbf{t}_1^\top \\ \mathbf{t}_2^\top \\ \dots \\ \mathbf{t}_n^\top \\ \mathbf{n}^\top \end{pmatrix} \cdot \nabla\varphi = \begin{pmatrix} \varphi_1 - \varphi_0 \\ \varphi_2 - \varphi_0 \\ \dots \\ \varphi_n - \varphi_0 \\ 0 \end{pmatrix}.$$

The projection or rotation of the edge vectors into the tangent plane of the vertex may lead to errors. This overdetermined system of linear equations is solved by the linear least squares method which constructs a square matrix by multiplying the matrix by its transpose, i.e.

$$\begin{pmatrix} \mathbf{t}_1^\top \\ \mathbf{t}_2^\top \\ \dots \\ \mathbf{t}_n^\top \\ \mathbf{n}^\top \end{pmatrix}^\top \cdot \begin{pmatrix} \mathbf{t}_1^\top \\ \mathbf{t}_2^\top \\ \dots \\ \mathbf{t}_n^\top \\ \mathbf{n}^\top \end{pmatrix} \cdot \nabla\varphi = \begin{pmatrix} \mathbf{t}_1^\top \\ \mathbf{t}_2^\top \\ \dots \\ \mathbf{t}_n^\top \\ \mathbf{n}^\top \end{pmatrix}^\top \cdot \begin{pmatrix} \varphi_1 - \varphi_0 \\ \varphi_2 - \varphi_0 \\ \dots \\ \varphi_n - \varphi_0 \\ 0 \end{pmatrix}.$$

B Discrete Geodesic and Normal Curvature

Geodesic and normal curvature are concepts of differential geometry [22]. They do not apply immediately to triangular meshes since they are based on the differential calculus, which delivers either trivial results (for points in the triangles) for triangular meshes, or cannot be applied at all (at edges or vertices of the mesh). However, the transfer of these concepts to triangular meshes has meanwhile become a topic of intensive research [14, 31, 49].

The *discrete curvature* κ at a point $\mathbf{p}_i$ of a polyline may be defined as the turning angle α_i of the polyline [43]. For polylines on a triangular surface mesh the curvature can be split into normal and geodesic curvature.

For the *discrete normal curvature* κ_{nd}, only the angles between the surface normals along the polyline are considered, while for the *geodesic curvature* κ_{gd}, the turning angles in the tangent plane of the polyline vertices are used. For vertices inside a triangle, the tangent plane is the plane of the triangle. For polyline vertices on a mesh edge, one of the two incident triangles has to be rotated into the plane of the other one, and the angle in the resulting plane is considered. If a polyline vertex is a mesh vertex $\mathbf{v}$, then the total angle around that vertex may differ from 2π, which results in the definition

$$\kappa_{gd}(\mathbf{v}) = \frac{2\pi}{\theta(\mathbf{v})}\left(\frac{\theta(\mathbf{v})}{2} - \beta(\mathbf{v})\right),$$

of the discrete geodesic curvature [86], with the total angle θ, and β being the left or the right polyline angle.

According to do Carmo [23], the geodesic curvature of an isoline of a scalar function may be calculated directly from the scalar function φ by

$$\kappa_g = -\mathrm{div}\left(\frac{\nabla\varphi}{\|\nabla\varphi\|}\right), \tag{B.1}$$

with div being the divergence of a vector field, i.e. the sum of the partial derivatives. This is nearly the same formula as for the Laplace operator, just with a normalized vector field. If φ is a smooth distance function then the Eikonal equation $\|\nabla\varphi\| = 1$ is valid.

A discrete version of Eq. B.1 on a triangular mesh may be derived analogously to a discrete version of the Laplace-Beltrami operator in Appendix A of Meyer et al. [79], as follows. The mean of $\kappa_g(\mathbf{m})$ in an environment $M(\mathbf{m})$ of a surface point $\mathbf{m}$ with an area of $A(\mathbf{m})$ is considered. Using Gauss' law on surfaces the surface integral is transformed to a contour integral,

$$\begin{aligned}\kappa_g(\mathbf{m}) &= \lim_{A(\mathbf{m})\to 0} \frac{1}{A(\mathbf{m})} \iint_{M(\mathbf{m})} \mathrm{div}\left(\frac{\nabla\varphi}{\|\nabla\varphi\|}\right) \mathrm{d}M \\ &= \lim_{A(\mathbf{m})\to 0} \frac{1}{A(\mathbf{m})} \oint_{\partial M(\mathbf{m})} \frac{\nabla\varphi^\top}{\|\nabla\varphi\|} \cdot \mathbf{n}\,\mathrm{d}L\,.\end{aligned}$$

Transferring the contour integral to an environment of every mesh vertex and using the discrete gradient of Eq. A.1 results in a sum which defines a discrete approximation of $\kappa_g(\mathbf{m})$

for every vertex $\mathbf{v}_i$. The mean curvature $\bar{\kappa}_g(\mathbf{v}_i)$ in the Voronoi area $A(\mathbf{v}_i)$ of vertex $\mathbf{v}_i$ is then

$$\begin{aligned}
\bar{\kappa}_g(\mathbf{v}_i) &= \frac{1}{A(\mathbf{v}_i)} \sum_{(i,j,k)\in N_f(i)} \frac{\nabla\varphi^\top}{\|\nabla\varphi\|} \frac{1}{2}(\mathbf{v}_j - \mathbf{v}_k)^\perp \\
&= \frac{1}{A(\mathbf{v}_i)} \sum_{(i,j,k)\in N_f(i)} (\varphi_j - \varphi_i) \cdot \frac{((\mathbf{v}_i - \mathbf{v}_k)^\perp)^\top \cdot (\mathbf{v}_j - \mathbf{v}_k)^\perp}{4A_T\|\nabla\varphi\|} \\
&\quad + (\varphi_k - \varphi_i) \cdot \frac{((\mathbf{v}_j - \mathbf{v}_i)^\perp)^\top \cdot (\mathbf{v}_j - \mathbf{v}_k)^\perp}{4A_T\|\nabla\varphi\|} \\
&= \frac{1}{A(\mathbf{v}_i)} \sum_{(i,j,k)\in N_f(i)} \frac{(\varphi_j - \varphi_i)\cdot \cot\angle(\mathbf{v}_k) + (\varphi_k - \varphi_i)\cdot \cot\angle(\mathbf{v}_j)}{2\|\nabla\varphi\|},
\end{aligned}$$

with

$$\|\nabla\varphi\| = \left\|(\varphi_j - \varphi_i)\cdot \frac{(\mathbf{v}_i - \mathbf{v}_k)^\perp}{2A_T} + (\varphi_k - \varphi_i)\cdot \frac{(\mathbf{v}_j - \mathbf{v}_i)^\perp}{2A_T}\right\|,$$

and $N_f(i)$ the set of triangles incident to vertex $\mathbf{v}_i$.

C Medial Axis and Voronoi Diagram Calculation on Triangular Meshes

For a source set S of points on a mesh surface, the medial axis and the Voronoi diagram can be calculated from the data structure delivered by the MMP algorithm. In the original paper of Mitchell et al. [80] the distance function is interpreted as an (implicit) Voronoi diagram, because it can answer the question which the nearest source is. Liu et al. [73] have explicitly presented an algorithm which derives a Voronoi diagram from a geodesic distance function on a triangular mesh by using the MMP algorithm. After preprocessing the mesh, the algorithm identifies triangles containing Voronoi vertices and traces the Voronoi edges from there. However, it ignores that Voronoi vertices can have more than three incident Voronoi edges and that Voronoi vertices can lie on mesh vertices or edges. It also ignores that Voronoi edges can go through mesh vertices, which invalidates their triangle classification.

The alternative approach proposed in Alg. 5 remedies those limitations.

Algorithm 5 Voronoi diagram calculation

```
1: Approximate input source set as sites
2: for all site ids $i$ do
3:     Calculate Voronoi edges for Voronoi cell $i$ (cf. Alg. 6)
4: end for
5: Identify corresponding Voronoi vertices
6: Remove obsolete Voronoi edges between neighboring sites
```

Its first step is the execution of the MMP algorithm, with the slight modification that the windows on the edges have to store one additional information which is the site id of the nearest site. The reason is that all windows with a specific site id induce a Voronoi cell.

Next, all face sets F_i whose faces contain a window with the specified site id i and a window with another site id are identified. The Voronoi edges surrounding the Voronoi cell of site i pass through the face set F_i.

Then, for every face in F_i the intersection points are calculated. Intersections along the edges of a face are identified by a change of site id of two consecutive windows. If the site id changes from one edge to the next, then a Voronoi edge passes through the vertex in-between.

There are two cases to distinguish related to the number of intersections. The first case is that only two intersection points are in a face, which are then connected by a line segment or by a polyline, if the face contains Voronoi vertices. The second case occurs when at least four intersection points are in a triangle. Connecting these would be rather simple if all Voronoi cells were convex, but unfortunately this is not the case. To solve this problem the triangle is adaptively subdivided until just two intersection points are left in each triangle, which leads again to case one.

The next construction step takes care of the faces which contain Voronoi vertices. Faces that are part of more than two Voronoi cells can contain Voronoi vertices. The number of vertices and their position is again found by subdividing the triangle. The 2D position $\mathbf{x}$

Algorithm 6 Voronoi edges for Voronoi cell i

```
1: Identify faces $F_i$ which contain a window of site $i$ and a window of another site
2: for all face $f$ in $F_i$ do
3:     Calculate edge intersections for face $f$
4:     if face $f$ has more than two intersections then
5:         Recursively call this algorithm for a subdivided triangle
6:     else
7:         if number of ids in face $f > 2$ then
8:             Calculate Voronoi vertex position (Eq. C.1)
9:             if incoming Voronoi edges do not intersect in a single vertex then
10:                Recursively call this algorithm for a subdivided triangle
11:            else
12:                Connect the two intersections with the vertex by two line segments
13:                Set neighbors of the two line segments to the resp. nearest site id
14:            end if
15:        else
16:            Connect the two edge intersection points by a line segment
17:            Set neighbor of line segment to other site id in $f$
18:        end if
19:    end if
20: end for
21: Sort and segment line segments into Voronoi edges
22: return Voronoi edges as polylines with neighbor ids
```

of a Voronoi vertex is determined by solving a system of linear equations, which results from minimizing the distance to all incoming Voronoi edges with intersection points $\mathbf{p}_i$ and normals $\mathbf{n}_i$, i.e.

$$\sum_i ((\mathbf{x} - \mathbf{p}_i) \cdot \mathbf{n}_i)^2 \to \min . \tag{C.1}$$

The normal direction $\mathbf{n}_i$ of a Voronoi edge, which has to be known for these equations, is the bisector of the two adjacent shortest path directions. These shortest path directions are stored in each window on the edges. A triangle containing a vertex is only subdivided if the intersection point deviates from the lines by a given threshold. This results in a fast adaptive calculation of Voronoi vertices with a predefined precision.

Special cases, where a Voronoi vertex lies on a mesh edge or a mesh vertex are reduced to the case where the Voronoi vertex lies on a mesh vertex by subdivision. When a Voronoi vertex lies on a mesh vertex, then the mesh can be subdivided such that the resulting triangles are only part of two different Voronoi cells, so that a calculation of the Voronoi Vertex position becomes obsolete.

The information about the neighboring site is also extracted from the distance function for each line segment, which is generated by Alg. 6.

The next step is to sort and segment the boundary of each Voronoi cell into Voronoi edges. After all Voronoi cell boundaries are calculated, corresponding Voronoi vertices are identified by comparing the vertex position on the mesh and along the Voronoi cell boundaries. Then, the cell boundaries are merged into a single Voronoi diagram and a graph representation containing the Voronoi vertices and edges is established.

Nomenclature

Basics

$\mathbf{a} = (a_1, \ldots, a_n)^\top$	(column-)vector $\mathbf{a}$ with elements a_i
$\mathbf{a}^\top = (a_1, \ldots, a_n)$	transpose of vector $\mathbf{a}$ with elements a_i
$\mathbf{a}^\top = (a_1 \ldots a_n)$	alternative notation for row vectors
$\mathbf{A} = (A_{ij})$	matrix $\mathbf{A}$ with elements A_{ij}
$d(\mathbf{a}, \mathbf{b})$	distance from $\mathbf{a}$ to $\mathbf{b}$ under metric d
$\|\mathbf{a}\|_d$	norm of vector $\mathbf{a}$ in metric d

Tool Pose Optimization

$\mathbf{p}$	point on surface
$\mathbf{q}$	gun reference point
$\mathbf{a}$	spray direction
f	flow rate distribution function
h	coating height
V_T	volume flow rate [mm^3/s]
V_M	coating height density
$\mathbf{m}$	gun base point
M	manifold M
t	time
t_M	spray time density [s/mm^2]
h_M	coating height density [mm/mm^2]
$h^s_{\mathbf{p}}$	desired coating height
$\mathcal{G}$	set of gun configurations
$\mathcal{P}$	set of height sample points
ρ_{mass}	mass density
ρ_j	gun sampling density at gun base point $\mathbf{m}_j$
Δ_M	Laplace-Beltrami operator
Z	objective function

C	coefficients
d_s	desired distance
$v_{j,\text{int}}$	interpolation of $1/t_M$
v_s	desired value for $1/t_M$
$d_{j,\text{int}}$	interpolated distance

Tool-adaptive Impact Paths

S	source set
$\mathbf{s}$	point in source set
$\mathbf{m}$	point on M
d_S	distance function for S
$\mathbf{n}$	normal vector
$D_{\mathbf{p}}$	metric tensor
$\mathbf{t}$	tangent
$K_{\mathbf{p}}$	curvature tensor
$\mathbf{w}$	path sample point
$v_{\mathbf{w}}$	velocity
ω	path interval
κ_i	principal curvatures

Quantitative Improvement of Tool Impact Paths

$d_{S,\sigma}$	distance function with offsets
I_d	isoline with distance d
$\mathcal{A}_S$	medial axis for S
$\widehat{\mathcal{A}}_S$	pruned medial axis
R	region
$\widehat{R}$	region for pruned medial axis
$\widehat{S}$	zero-isoline for pruned medial axis
w	medial axis weight function
κ_g	geodesic curvature
κ_{gd}	discrete geodesic curvature
κ_{nd}	discrete normal curvature

f	objective function
c	coefficients

Manipulator Placement Optimization

${}^{\mathrm{D}}\mathbf{T}_{\mathrm{C}}$	coordinate transformation from frame C to frame D
${}^{\mathrm{D}}\mathbf{R}_{\mathrm{C}}$	rotation from frame C to frame D
$\mathbf{o}_{\mathrm{C}}$	origin of frame C
W	world reference frame
P	part reference frame
M	manipulator reference frame
T	tool reference frame
E_{M}	end-effector reference frame
$\boldsymbol{\varphi}$	configuration vector
$\boldsymbol{\varphi}(t)$	manipulator path
$\boldsymbol{\omega}(t)$	angular velocity
$\boldsymbol{\alpha}(t)$	angular acceleration
t_f	cycle-time or duration
$\mathbf{x}$	state vector
$\hat{\mathbf{x}}$	reduced state vector
$\mathbf{u}$	control vector
$\mathbf{J}$	Jacobian matrix
$\sigma_{\min}$	smallest singular value
$\sigma_{\max}$	largest singular value
M	basis function
A_T	triangle area
$\mathbf{h}_s$	state and control constraints
$\mathbf{h}_a$	application constraints
$\mathbf{h}_c$	collision constraints
$\mathbf{h}_p$	placement constraints
F	objective function
L	Lagrangian function
λ	coefficients

Bibliography

[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun. Anisotropic polygonal remeshing. In: *ACM SIGGRAPH*, ACM, New York, NY, USA, 2003, pp. 485–493. DOI: 10.1145/1201775.882296.

[2] J. K. Antonio. Optimal trajectory planning for spray coating. In: *IEEE International Conference on Robotics and Automation*, 1994, pp. 2570–2577. DOI: 10.1109/ROBOT.1994.351125.

[3] P. J. Arrazola, T. Özel, D. Umbrello, M. Davies, and I. S. Jawahir. Recent advances in modelling of metal machining processes. *CIRP Annals - Manufacturing Technology*, 2013, **62**(2):695–718. DOI: 10.1016/j.cirp.2013.05.006.

[4] N. A. Aspragathos and S. Foussias. Optimal location of a robot path when considering velocity performance. *Robotica*, 2002, **20**(2):139–147. DOI: 10.1017/S0263574701003708.

[5] A. A. Ata. Optimal trajectory planning of manipulators: a review. *Journal of Engineering Science and Technology*, 2007, **2**(1):32–54.

[6] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi. Hierarchical segmentation of surfaces embedded in $\mathbb{R}^3$ for auto-body painting. In: *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005, pp. 574–579. DOI: 10.1109/ROBOT.2005.1570179.

[7] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi. Uniform coverage of automotive surface patches. *International Journal of Robotics Research*, 2005, **24**(11):883–898. DOI: 10.1177/0278364905059058.

[8] D. Barral, J.-P. Perrin, E. Dombre, and A. Liégeois. Simulated annealing combined with a constructive algorithm for optimising assembly workcell layout. *The International Journal of Advanced Manufacturing Technology*, 2001, **17**:593–602. DOI: 10.1007/s001700170143.

[9] J. T. Betts. Practical Methods for Optimal Control and Estimation Using Nonlinear Programming. 2nd ed. SIAM, 2010, DOI: 10.1137/1.9780898718577.

[10] H.-G. Beyer and H.-P. Schwefel. Evolution strategies – a comprehensive introduction. *Natural Computing*, 2002, **1**(1):3–52. DOI: 10.1023/A:1015059928466.

[11] D. Biermann, E. Krebs, A. Sacharow, and P. Kersting. Using NC-path deformation for compensating tool deflections in micromilling of hardened steel. *Procedia CIRP*, 2012, **1**:132–137. DOI: 10.1016/j.procir.2012.04.022.

[12] M. B. Bieterman and D. R. Sandstrom. A curvilinear tool-path method for pocket machining. *Journal of Manufacturing Science and Engineering*, 2003, **125**(4):709–715.

[13] H. Blum. A transformation for extracting new descriptors of shape. In: *Models for the Perception of Speech and Visual Form*, ed. by W. Wathen-Dunn. Cambridge: MIT Press, 1967, pp. 362–380.

[14] A. I. Bobenko, P. Schröder, J. M. Sullivan, and G. M. Ziegler, eds. Discrete Differential Geometry. Birkhäuser, 2008.

[15] D. Bommes and L. Kobbelt. Accurate computation of geodesic distance fields for polygonal curves on triangle meshes. *VMV*, 2007, pp. 151–160.

[16] G. Boschetti, R. Rosa, and A. Trevisani. Optimal robot positioning using task-dependent and direction-selective performance indexes: general definitions and application to a parallel robot. *Robotics and Computer-Integrated Manufacturing*, 2013, **29**(2):431–443. DOI: 10.1016/j.rcim.2012.09.013.

[17] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy. Polygon Mesh Processing. AK Peters, 2010.

[18] M. Bouard, V. Pateloup, and P. Armand. Pocketing toolpath computation using an optimization method. *Computer-Aided Design*, 2011, **43**(9):1099–1109. DOI: 10.1016/j.cad.2011.05.008.

[19] G. E. Box and D. W. Behnken. Some new three level designs for the study of quantitative variables. *Technometrics*, 1960, **2**(4):455–475.

[20] R. N. Bracewell. The Fourier Transformation and Its Applications. 3rd ed. McGraw-Hill, 2000.

[21] W. Bu, Z. Liu, and J. Tan. Industrial robot layout based on operation sequence optimisation. *International Journal of Production Research*, 2009, **47**(15):4125–4145.

[22] M. P. do Carmo. Differential Geometry of Curves and Surfaces. Prentice-Hall, 1976.

[23] M. P. do Carmo. Riemannian Geometry. Birkhäuser, 1992.

[24] H. Chen, T. Fuhlbrigge, and X. Li. A review of CAD-based robot path planning for spray painting. *Industrial Robot: An International Journal*, 2009, **36**(1):45–50. DOI: 10.1108/01439910910924666.

[25] C.-J. Chiou and Y.-S. Lee. A machining potential field approach to tool path generation for multi-axis sculptured surface machining. *Computer-Aided Design*, 2002, **34**(5):357–371. DOI: 10.1016/S0010-4485(01)00102-6.

[26] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Boston, 2005.

[27] H. Chou and J. Sadler. Optimal location of robot trajectories for minimization of actuator torque. *Mechanism and Machine Theory*, 1993, **28**(1):145–158. DOI: 10.1016/0094-114X(93)90053-X.

[28] D. C. Conner, A. Greenfield, P. N. Atkar, A. Rizzi, and H. Choset. Paint deposition modelling for trajectory planning on automotive surfaces. *IEEE Transactions on Automotive Science and Engineering*, 2005, **2**:381–392. DOI: 10.1109/TASE.2005.851631.

[29] J. G. van der Corput. Verteilungsfunktionen I. *Akademie van Wetenschappen*, 1935, **38**:813–821.

[30] J. R. Davis, ed. Handbook of Thermal Spray Technology. ASM International, 2004.

[31] M. Desbrun, E. Kanso, and Y. Tong. Discrete differential forms for computational modeling. In: *ACM SIGGRAPH 2005 Courses*, ACM, Los Angeles, California, 2005, DOI: 10.1145/1198555.1198666.

[32] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999, DOI: 10.1145/311535.311576.

[33] S. Dhanik and P. Xirouchakis. Contour parallel milling tool path generation for arbitrary pocket shape using a fast marching method. *The International Journal of Advanced Manufacturing Technology*, 2010, **50**(9–12):1101–1111. DOI: 10.1007/s00170-010-2580-z.

[34] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1959, **1**(1):269–271. DOI: 10.1007/BF01386390.

[35] S. Duncan, P. Jones, and P. Wellstead. A frequency-domain approach to determining the path separation for spray coating. *IEEE Transactions on Automation Science and Engineering*, 2005, **2**(3):233–239. DOI: 10.1109/TASE.2005.850393.

[36] G. Elber, E. Cohen, and S. Drake. MATHSM: medial axis transform toward high speed machining of pockets. *Computer-Aided Design*, 2005, **37**:241–250.

[37] G. Elber, E. Cohen, and S. Drake. C^1 continuous toolpath generation toward 5-axis high speed machining. *Computer-Aided Design and Applications*, 2006, **3**(6):803–810.

[38] B. Fardanesh and J. Rastegar. Minimum cycle time location of a task in the workspace of a robot arm. In: *Proceedings of the 27th IEEE Conference on Decision and Control*, 1988, pp. 2280–2283. DOI: 10.1109/CDC.1988.194742.

[39] P. Fauchais, M. Fukumoto, A. Vardelle, and M. Vardelle. Knowledge concerning splat formation: an invited review. *Journal of Thermal Spray Technology*, 2004, **13**(3):337–360. DOI: 10.1361/10599630419670.

[40] R. Featherstone and D. E. Orin. Dynamics. In: *Springer Handbook of Robotics*, ed. by B. Siciliano and O. Khatib. Springer Berlin Heidelberg, 2008, pp. 35–65, DOI: 10.1007/978-3-540-30301-5_3.

[41] J. Feddema. Kinematically optimal robot placement for minimum time coordinated motion. In: *IEEE International Conference on Robotics and Automation*, 1996, pp. 3395–3400. DOI: 10.1109/ROBOT.1996.509229.

[42] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. HEALPix: a framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 2005, **622**(2):759–771. DOI: 10.1086/427976.

[43] E. Grinspun and A. Secord. Introduction to discrete differential geometry: the geometry of plane curves. In: *ACM SIGGRAPH 2006 Courses*, ACM, Boston, Massachusetts, 2006, pp. 1–4. DOI: 10.1145/1185657.1185659.

[44] F. L. Hammond and K. Shimada. Improvement of manufacturing workcell layout design using weighted isotropy metrics. In: *IEEE International Conference on Mechatronics and Automation*, 2009, pp. 3408–3414. DOI: 10.1109/ICMA.2009.5246361.

[45] N. Hansen. The CMA evolution strategy: a comparing review. In: *Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms*, ed. by J. A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea. Springer, 2006, pp. 75–102.

[46] D. Hegels and H. Müller. Evolutionary path generation for reduction of thermal variations in thermal spray coating. In: *Proceedings GECCO 2013 (Genetic and Evolutionary Computation Conference 2013)*, ACM, New York, 2013, pp. 1277–1284.

[47] M. Held and C. Spiegelberger. A smooth spiral tool path for high speed machining of 2D pockets. *Computer-Aided Design*, 2009, **41**(7):539–550.

[48] J. Hemmerle and F. Prinz. Optimal path placement for kinematically redundant manipulators. In: *IEEE International Conference on Robotics and Automation*, 1991, pp. 1234–1244. DOI: 10.1109/ROBOT.1991.131781.

[49] A. N. Hirani. Discrete exterior calculus, PhD thesis, Pasadena, California: California Institute of Technology, May 2003.

[50] I. Hotz, J. Sreevalsan-Nair, H. Hagen, and B. Hamann. Tensor field reconstruction based on eigenvector and eigenvalue interpolation. In: *Scientific Visualization: Advanced Concepts*, 2010, pp. 110–123.

[51] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 2007, **15**(1):1–28. DOI: 10.1162/evco.2007.15.1.1.

[52] C. Igel, V. Heidrich-Meisner, and T. Glasmachers. Shark. *Journal of Machine Learning Research*, 2008, **9**:993–996.

[53] C. Igel, T. Suttorp, and N. Hansen. Steady-state selection and efficient covariance matrix update in the multi-objective CMA-ES. In: *Fourth International Conference on Evolutionary Multi-Criterion Optimization*, Springer-Verlag, Matsushima, Japan, 2007, pp. 171–185.

[54] M. Jones, J. Bærentzen, and M. Sramek. 3D distance fields: a survey of techniques and applications. *Visualization and Computer Graphics, IEEE Transactions on*, 2006, **12**(4):581–599. DOI: 10.1109/TVCG.2006.56.

[55] P. D. A. Jones, S. R. Duncan, T. Rayment, and P. S. Grant. Optimal robot path for minimizing thermal variations in a spray deposition process. *IEEE Transactions on Control Systems Technology*, 2007, **15**(1):1–11.

[56] B. Kamrani, V. Berbyuk, D. Wäppling, U. Stickelmann, and X. Feng. Optimal robot placement using response surface method. *International Journal of Advanced Manufacturing Technology*, 2009, **44**(1-2):201–210. DOI: 10.1007/s00170-008-1824-7.

[57] L. Kavan, S. Collins, and J. Zara. *Dual quaternions for rigid transformation blending*, tech. rep., 2006.

[58] T. Kim. Constant cusp height tool paths as geodesic parallels on an Riemannian manifold. *Computer-Aided Design*, 2007, **39**(6):477–489.

[59] T. Kim and S. E. Sarma. Optimal sweeping paths on a 2-manifold: a new class of optimization problems defined by path structures. *IEEE Transactions of Robotics and Automation*, 2003, **19**(4):613–636.

[60] R. Kimmel and A. M. Bruckstein. Shape offsets via level sets. *Computer-Aided Design*, 1993, **25**(3):154–162. DOI: 10.1016/0010-4485(93)90040-U.

[61] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences*, 1998, **95**(15):8431–8435.

[62] A. Kout and H. Müller. Parameter optimization for spray coating. *Advances in Engineering Software*, 2009, **40**(10):1078–1086. DOI: 10.1016/j.advengsoft.2009.03.001.

[63] A. Kout and H. Müller. Quantitative improvement of tool impact paths defined by isolines of scalar functions on triangular mesh workpiece surfaces. *The International Journal of Advanced Manufacturing Technology*, 2014, **70**(1-4):237–255. DOI: 10.1007/s00170-013-5152-1.

[64] A. Kout and H. Müller. Tool-adaptive offset paths on triangular mesh workpiece surfaces. *Computer-Aided Design*, 2014, **50**:61–73. DOI: 10.1016/j.cad.2014.01.009.

[65] J. J. Kuffner. Effective sampling and distance metrics for 3D rigid body path planning. In: *International Conference on Robotics and Automation*, 2004, pp. 3993–3998. DOI: 10.1109/ROBOT.2004.1308895.

[66] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast distance queries with rectangular swept sphere volumes. In: *IEEE International Conference on Robotics and Automation (ICRA)*, 2000, pp. 3719–3726. DOI: 10.1109/ROBOT.2000.845311.

[67] A. Lasemi, D. Xue, and P. Gu. Recent development in CNC machining of freeform surfaces: a state-of-the-art review. *Computer-Aided Design*, 2010, **42**(7):641–654. DOI: 10.1016/j.cad.2010.04.002.

[68] I. Lazoglu. Sculpture surface machining: a generalized model of ball-end milling force system. *International Journal of Machine Tools and Manufacture*, 2003, **43**(5):453–462. DOI: 10.1016/S0890-6955(02)00302-4.

[69] D. T. Lee and R. L. Drysdale. Generalization of Voronoi diagrams in the plane. *SIAM Journal on Computing*, 1981, **10**(1):73–87. DOI: 10.1137/0210006.

[70] D. Levin. Mesh-independent surface interpolation. In: *Geometric Modeling for Scientific Visualization*, ed. by G. Brunnett, B. Hamann, H. Müller, and L. Linsen. Springer-Verlag, 2003, pp. 37–49.

[71] M. Lin and S. Gottschalk. Collision detection between geometric models: a survey. In: *Proc. of IMA Conference on Mathematics of Surfaces*, 1998, pp. 602–608.

[72] R.-S. Lin and Y. Koren. Efficient tool-path planning for machining free-form surfaces. *ASME Journal of Engineering for Industry*, 1996, **118**(1):20–28. DOI: 10.1115/1.2803642.

[73] Y. Liu, Z.-Q. Chen, and K. Tang. Construction of iso-contours, bisectors, and Voronoi diagrams on triangulated surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011, **33**(8):1502–1517.

[74] A. Lopes and E. S. Pires. Optimization of the workpiece location in a machining robotic cell. *International Journal of Advanced Robotic Systems*, 2011, **8**(6):37–46.

[75] L. N. López de Lacalle, A. Lamikiz, J. Sánchez, and M. Salgado. Toolpath selection based on the minimum deflection cutting forces in the programming of complex surfaces milling. *International Journal of Machine Tools and Manufacture*, 2007, **47**(2):388–400. DOI: 10.1016/j.ijmachtools.2006.03.010.

[76] L. López de Lacalle, A. Rodríguez, A. Lamikiz, A. Celaya, and R. Alberdi. Five-axis machining and burnishing of complex parts for the improvement of surface roughness. *Materials and Manufacturing Processes*, 2011, **26**(8):997–1003. DOI: 10.1080/10426914.2010.529589.

[77] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *ACM Siggraph Computer Graphics*, 1987, **21**(4):163–169.

[78] M. D. McKay, R. J. Beckman, and W. J. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 1979, **21**(2):239–245. DOI: 10.1080/00401706.1979.10489755.

[79] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In: *Visualization and Mathematics III*, ed. by H.-C. Hege and K. Polthier. Heidelberg: Springer-Verlag, 2003.

[80] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 1987, **16**(4):647–668.

[81] S. Mitsi, K.-D. Bouzakis, D. Sagris, and G. Mansour. Determination of optimum robot base location considering discrete end-effector positions by means of hybrid genetic algorithm. *Robotics and Computer-Integrated Manufacturing*, 2008, **24**(1):50–59. DOI: 10.1016/j.rcim.2006.08.003.

[82] A. Nektarios and N. A. Aspragathos. Optimal location of a general position and orientation end-effector's path relative to manipulator's base, considering velocity performance. *Robotics and Computer-Integrated Manufacturing*, 2010, **26**(2):162–173. DOI: 10.1016/j.rcim.2009.07.003.

[83] B. Nelson and M. Donath. Optimizing the location of assembly tasks in a manipulator's workspace. *Journal of Robotic Systems*, 1990, **7**(6):791–811. DOI: 10.1002/rob.4620070602.

[84] J. Nocedal and S. Wright. Numerical Optimization. Springer, 1999.

[85] J. Pamanes-García, E. Cuan-Durón, and S. Zeghloul. Single and multi-objective optimization of path placement for redundant robotic manipulators. *INGENIERÍA Investigación y Tecnología*, 2008, **9**(3):231–257.

[86] K. Polthier and M. Schmies. Straightest geodesics on polyhedral surfaces. In: *Mathematical Visualization*, ed. by H.-C. Hege and K. Polthier. Springer Verlag, 1998, pp. 135–150.

[87] R. Ramabhadran and J. K. Antonio. Fast solution techniques for a class of optimal trajectory planning problems with applications to automated spray coating. *IEEE Transactions on Robotics and Automation*, 1997, **13**(4):519–530. DOI: 10.1109/70.611308.

[88] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. CHOMP: gradient optimization techniques for efficient motion planning. In: *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, IEEE Press, Kobe, Japan, 2009, pp. 4030–4035.

[89] R. Ur-Rehman, S. Caro, D. Chablat, and P. Wenger. Multi-objective path placement optimization of parallel kinematics machines based on energy consumption, shaking forces and maximum actuator torques: application to the orthoglide. *Mechanism and Machine Theory*, 2010, **45**(8):1125–1141.

[90] R. Ur-Rehman, S. Caro, D. Chablat, and P. Wenger. Path placement optimization of manipulators based on energy consumption: application to the orthoglide 3-axis. *Transactions of the Canadian Society for Mechanical Engineering*, 2009, pp. 1–19.

[91] R. R. dos Santos, V. Steffen, and S. d. F. P. Saramago. Optimal task placement of a serial robot manipulator for manipulability and mechanical power optimization. *Intelligent Information Management*, 2010, **2**(9):512–525.

[92] J. A. Sethian. Level Set Methods and Fast Marching Methods. 2nd ed. Cambridge University Press, 1999.

[93] J. A. Sethian and A. Vladimirsky. Fast methods for the Eikonal and related Hamilton-Jacobi equations on unstructured meshes. *Proceedings of the National Academy of Sciences*, 2000, **97**(11):5699–5703.

[94] J. A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations: theory and algorithms. *SIAM Journal on Numerical Analysis*, 2003, **41**(1):325–363.

[95] M. Sharir. Intersection and closest-pair problems for a set of planar discs. *SIAM Journal on Computing*, 1985, **14**(2):448–468. DOI: 10.1137/0214034.

[96] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. In: *ACM SIGGRAPH Proc.* New York, NY, USA, 2005, pp. 553–560. DOI: 10.1145/1186822.1073228.

[97] T. Suttorp, N. Hansen, and C. Igel. Efficient covariance matrix update for variable metric evolution strategies. *Machine Learning*, 2009, **75**(2):167–197. DOI: 10.1007/s10994-009-5102-1.

[98] K.-F. Tchon, M. Khachan, F. Guibault, and R. Camarero. Three-dimensional anisotropic geometric metrics based on local domain curvature and thickness. *Computer-Aided Design*, 2005, **37**(2):173–187. DOI: 10.1016/j.cad.2004.05.007.

[99] J. Tierny and V. Pascucci. Generalized topological simplification of scalar fields on surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 2012, **18**(12):2005–2013. DOI: 10.1109/TVCG.2012.228.

[100] M. Togai. An application of the singular value decomposition to manipulability and sensitivity of industrial robots. *SIAM Journal on Algebraic Discrete Methods*, 1986, **7**(2):315–320. DOI: 10.1137/0607034.

[101] M.-J. Tsai. Workspace geometric characterization and manipulability of industrial robots, PhD thesis, Ohio State University, 1986.

[102] G. Turk. Re-tiling polygonal surfaces. *ACM SIGGRAPH Computer Graphics*, 1992, **26**(2):55–64. DOI: 10.1145/142920.134008.

[103] N. Vahrenkamp, T. Asfour, and R. Dillmann. Robot placement based on reachability inversion. In: *Robotics and Automation (ICRA), IEEE International Conference On*, 2013, pp. 1970–1975.

[104] M. Vincze, A. Pichler, G. Biegelbauer, K. Häusler, A. Andersen, O. Madsen, and M. Kristiansen. Automatic robotic spray painting of low volume high variant parts. In: *Proc. 33rd International Symposium on Robotics (ISR)*, 2002.

[105] G.-C. Vosniakos and E. Matsas. Improving feasibility of robotic milling through robot placement optimisation. *Robotics and Computer-Integrated Manufacturing*, 2010, **26**(5):517–525. DOI: 10.1016/j.rcim.2010.04.001.

[106] T. Voß, N. Hansen, and C. Igel. Improved step size adaptation for the MO-CMA-ES. In: *Proc. of the 12th Annual Conference on Genetic and Evolutionary Computation*, ACM, Portland, Oregon, USA, 2010, pp. 487–494. DOI: 10.1145/1830483.1830573.

[107] K. Waldron and J. Schmiedeler. Kinematics. In: *Springer Handbook of Robotics*, ed. by B. Siciliano and O. Khatib. Springer Berlin Heidelberg, 2008, pp. 9–33, DOI: 10.1007/978-3-540-30301-5_2.

[108] T. Weinkauf, Y. Gingold, and O. Sorkine. Topology-based smoothing of 2D scalar fields with C^1-continuity. *Computer Graphics Forum*, 2010, **29**(3):1221–1230. DOI: 10.1111/j.1467-8659.2009.01702.x.

[109] R. Weinstock. Calculus of Variations. Dover Publications, 1974.

[110] R. Weller and G. Zachmann. A unified approach for physically-based simulations and haptic rendering. In: *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games*, 2009, pp. 151–159.

[111] T. Wiederkehr, A. Kout, and H. Müller. Graphical simulation and visualization of spray coating processes in computer-aided engineering. In: *Vision, Modeling and Visualization*, Akademische Verlagsgesellschaft Aka GmbH, 2008, pp. 13–20.

[112] T. Yoshikawa. Foundations of Robotics: Analysis and Control. MIT Press, Cambridge, MA, USA, 1990.

[113] L. Zhang, Y. J. Kim, G. Varadhan, and D. Manocha. Generalized penetration depth computation. In: *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling*, ACM, Cardiff, Wales, United Kingdom, 2006, pp. 173–184. DOI: 10.1145/1128888.1128914.

[114] C. Zhuang, Z. Xiong, and H. Ding. High speed machining tool path generation for pockets using level sets. *International Journal of Production Research*, 2010, **48**(19):5749–5766. DOI: 10.1080/00207540903232771.