The International Journal of Advanced Manufacturing Technology (2021) 115:275–289
https://doi.org/10.1007/s00170-021-07001-w

ORIGINAL ARTICLE

# GPU accelerated voxel-based machining simulation

Florian Schnös[1] · Dirk Hartmann[2] · Birgit Obst[2] · Glenn Glashagen[1]

Received: 9 August 2020 / Accepted: 25 March 2021 / Published online: 8 May 2021


**Abstract**
The simulation of subtractive manufacturing processes has a long history in engineering. Corresponding predictions are utilized for planning, validation and optimization, e.g., of CNC-machining processes. With the up-rise of flexible robotic machining and the advancements of computational and algorithmic capability, the simulation of the coupled machine-process behaviour for complex machining processes and large workpieces is within reach. These simulations require fast material removal predictions and analysis with high spatial resolution for multi-axis operations. Within this contribution, we propose to leverage voxel-based concepts introduced in the computer graphics industry to accelerate material removal simulations. Corresponding schemes are well suited for massive parallelization. By leveraging the computational power offered by modern graphics hardware, the computational performance of high spatial accuracy volumetric voxel-based algorithms is further improved. They now allow for very fast and accurate volume removal simulation and analysis of machining processes. Within this paper, a detailed description of the data structures and algorithms is provided along a detailed benchmark for common machining operations.

**Keywords** Virtual machining · GPU · Voxel · Milling · Robot

## 1 Introduction

Simulation of material removal processes is a well-established practice in industry. It is mostly used in the context of manufacturing planning, machining process simulation and optimization, e.g. for the selection of the right tools for a given design and material or the definition of the tool path including appropriate process parameters, like feed rate, spindle speed or depth of cut. Furthermore, recently novel concepts for the combination of material removal simulation and dynamic machine tool simulation have shown significant improvements for the compensation of deflections and process stability prediction [31]. In particular, a compensation of tool centre point deflections caused by process forces allows to increase the quality of produced parts, and therefore enable the utilization of low stiffness robotic milling systems for the machining under high process force loads. However, for a broad industrial application, fast and accurate physical machining process simulation is required to make robotic machining to become a reality [8, 22, 40].

Within this contribution, we demonstrate a novel concept for geometric material removal simulation and analysis, which allows for a fast and accurate prediction of the tool engagement and the process forces as a key enabler for industrialization of robotic machining, especially for large workpieces. The concept leverages recent advancements of voxel-based modelling on Graphic Processing Units (GPUs) [28] for the simulation of the material removal and the prediction of the engagement between the tool and the workpiece. Based on the geometric predictions, process forces can be calculated accurately using mechanistic process force models. By adopting the technology for machine tool design, milling operation planning, and feed

✉ Florian Schnös
florian.schnoes@tum.de

Dirk Hartmann
hartmann.dirk@siemens.com

Birgit Obst
birgit.obst@siemens.com

Glenn Glashagen
glenn.glashagen@tum.de

[1] Technical University of Munich, Boltzmannstr. 15, 85748, Garching, Germany

[2] Siemens Technology, Otto-Hahn-Ring 6, 81739, München, Germany

forward displacement compensation, a paradigm shift from rigidly designed machines to intelligent machine tools that adapt to expected process forces and performance criteria could be achieved.

Besides the use case of process force prediction, highly efficient and modifiable volumetric models with a high spatial resolution, a low memory footprint, efficient data structures, and data modification algorithms can be utilized in other areas like simulated surgery, process simulation for additive manufacturing, and simulation of modifiable environments.

### 1.1 Material removal simulation

In the past years, a number of methods have been proposed for material removal simulation, e.g. [21, 37]. An overview of existing volumetric models is shown in Fig. 1. These range from explicit geometry representation via simple faceted [30] or complex parameterized direct geometry representations [17] to spatially hierarchical structured descriptions.

The most prominent today are probably dexel-based methods, which are widely adopted by industry (e.g. [25]). These offer a good compromise between versatility as well as computational complexity, albeit having an-isotropic spatial resolution and potentially showing defects for sharp geometric features like corners and edges [1].

Most concepts for geometric representations originate from the field of computer graphics. In the recent years, computational geometry representation via voxels has found quite some interest in the computer graphics community, e.g. powerful and complex open-source libraries are available [26, 28]. In particular, voxel-based methods allow, due to their simple data structures, very efficient massive parallelization on GPUs [28]. Thus, extremely complex simulations can be performed on desktop computers where otherwise small clusters would be needed.

Leveraging GPUs allows an unbeaten performance per energy, costs, or even spatial requirements. Therefore, in the last years, one could observe how corresponding methods are being more and more adopted in industrial use.[1] For example, corresponding methods, used for early design simulation and optimization [16], also found their way into machining simulation [13, 37].

### 1.2 Tri-dexel and voxel models

Recent publications in the field of high performance [7], or high accuracy [19] simulation of material removal processes typically choose a tri-dexel representation of the mutable in-process geometry of the workpiece. Within the field of material removal simulation, the voxel representation is a less popular choice due to the perceived drawbacks in computational and memory efficiency of the voxelized workpiece.

While tri-dexel models being widely adopted and utilized in both industry and research, certain limitations apply to tri-dexel models. The number of the required dexels for the tri-dexel representation of a convex solid is proportional to the axis aligned projection of the surface area for an orthogonally arranged dexel grid [1]. Thus, the memory consumption and computational complexity is $\mathcal{O}(n^2)$, where $n$ represents the number of dexels in each dimension [1]. This leads to a practical limit of the dexel grid resolution. 400 dexel subdivisions per dimension are stated as unsuitable for real-time applications [27]. The simulation of the manufacturing of a submodel of a blade of an impeller, with 1024 dexel subdivisions per dimension, lead to simulation times beyond 1 h [23]. While the achievable subdivisions per dimension are sufficient for the simulation of macroscopic material removal for typical workpiece sizes, the limitations are evident for the simulation of the uncut chip geometry for large workpieces.

The number of required voxels for the representation of a solid is proportional to the volume of the axis aligned bounding box of the solid, if the voxels are arranged in a regular grid [1]. By spatial compression mechanisms, the memory consumption and computation efficiency of the naive dense voxel representation can be improved [7]. Octree data structures and the model presented in [31] are examples for the compression of homogeneous sub-volumes. Under the assumption of a perfect surface voxelization with implicit voxel position encoding and neglecting the memory consumption of the spatially compressed inner volume and necessary data structures, the voxel representation memory consumption would be proportional to the surface area of the solid. While these assumptions cannot be realized in actual implementations, the model presented in [31] comes close, by increasing the subdivision rate in the spatial voxel structure and balancing the memory consumption of tree structure and actual voxel data.

Thus, the two volumetric representations, exhibiting a memory consumption approximately proportional to the surface area of the solid, are competing in this scenario. Therefore, let us take a more quantitative comparison. While a voxel encodes one surface point of the solid, the dexel encodes an entrance and an exit point on the surface of the solid. For the encoding of the volumetric data of a voxel, ideally one bit is required. This bit indicates if the voxel is within or outside of the solid. The encoding of one dexel segment minimally includes the start and end point. For single precision data types, this results in 64 bits of data, if

[1] beyond machine learning applications which is probably the most common industrial use today

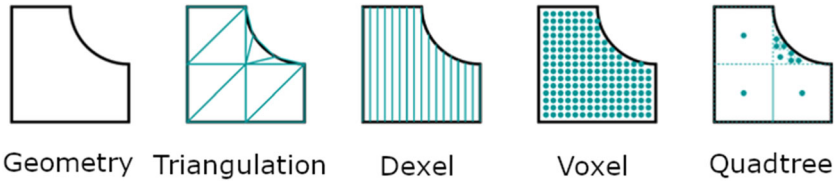


**Fig. 1** Different methods for material removal simulation. Within this contribution we will focus on dexel and voxel representations

typical additional information like surface normals and data pointers to the next dexel segment are neglected.

A cuboid with a size of $170 \times 120 \times 25\,\text{mm}^3$, a volume of $510{,}000\,\text{mm}^3$, and a surface area of $55{,}300\,\text{mm}^2$ shall be represented by three hypothetical models. A spatial voxel grid resolution of 0.1 mm and a dexel grid resolution of 0.566 mm is assumed. The memory consumption of the hypothetical models is stated in Table 1. Despite the assumptions for both the voxel and the dexel representation of a simple solid, the simplified thought experiment shows, that optimized voxel data structures can compete with dexel representations in terms of memory size per grid resolution. Both, the hypothetical perfect voxel shell and the dexel model require the same amount of memory, while the voxel grid resolution is 5.66 times higher.

The actual implementations of voxel and dexel models include additional data, which is, e.g., used to represent the tree structure of the hierarchical voxel data, multi-layer voxel shells, the surface normals of the dexel segments, and the structure of the dexel data. Furthermore, typical workpieces of interest have far more complex geometries, e.g. have varying sizes and complex features such as undercuts, sharp edges, sharp corners, and curved surfaces. Still our quantitative comparison should hold roughly. Within this contribution, state-of-the-art spatially compressed voxel model implementations on a CPU and a GPU will be compared to a widely adopted commercial dexel model implementation for the representation of a variety of in-process workpieces for machining simulations. The results will be evaluated in terms of computation time and model accuracy.

### 1.3 Our contribution

We focus on the highly parallel implementation of voxel-based volumetric models with high performance and low memory footprint. The approach is inspired by [28] and described in detail in Section 2. The provided technology can be used for various applications, which require large volumetric models with mutable geometry, high spatial resolution and high mutation performance. The feasibility of the developed model is demonstrated for the use case of machining simulation. By leveraging the decoupling of material removal simulation and force prediction, the performance of the virtual machining simulation described in [31] can be improved, while maintaining the benefits of a highly accurate and inherently consistent volume representation.

Based on a set of benchmark workpieces, the performance of the concept is compared with a sequential CPU implementation. Furthermore, the quality of the results of our voxel model is compared with a state-of-the-art dexel model available in NX CAM [34].

Finally, we show how this technology will allow large-scale robotic machining along a range of selected examples in Section 4.

The article closes with a discussion (Section 5) of the use cases and potentials of the presented voxel model and the underlying parallelization strategy. By leveraging emerging cloud infrastructures, formerly highly demanding volumetric simulations can become accessible for cost optimized local control systems and therefore provide novel simulation-based optimization and control strategies.

## 2 Process force and material removal simulation

This section introduces the concepts of the highly parallel implementation of voxel models for the representation of finite volumes intended for the usage in machining simulations.

In general, the changes of the engagement of a milling tool and the work piece are slow compared to the changes of the instantaneous cutting edge position and orientation, i.e. the rotation of the tool. Thus, we can decouple

**Table 1** Comparison of the required memory size for three hypothetical volumetric models

| Model | Grid resolution | Number of primitives | Size of primitive | Memory size |
|---|---|---|---|---|
| Naive voxel | 0.1 mm | $510{,}000\,\text{mm}^3/0.001\,\text{mm}^{-3} = 510{,}000{,}000$ | 1 bit | 60.8 MB |
| Perfect voxel shell | 0.1 mm | $55{,}300\,\text{mm}^2/0.01\,\text{mm}^{-2} = 5{,}530{,}000$ | 1 bit | 0.659 MB |
| Dexel | 0.566 mm | $0.5 \cdot 55{,}300\,\text{mm}^2/0.32\,\text{mm}^{-2} = 86{,}406$ | 64 bit | 0.659 MB |

the simulation of material removal from the process force simulation for machining operations. By separating the required high spatial resolution for the engagement simulation from the required high temporal resolution for the simulation of dynamic cutting forces, the overall process is optimized (see Fig. 2).

For the simulation of process forces (Section 2.4), the time-dependent tool engagement and instantaneous uncut chip thicknesses have to be simulated. The necessary information for these simulations is the changing work piece geometry (Section 2.2), the tool geometry (Section 2.3) and the tool path. Based on this information, engagement histograms are generated. In a subsequent step, the process forces are simulated based on a mechanistic cutting force model [31] (Section 2.5). The overall computational efficiency depends on the computational demanding volume removal simulation. The accuracy of the engagement generated histograms defines the accuracy of the process force predictions. The second step, the calculation of process forces, is computationally much less expensive and could, e.g., be performed on an edge device while the first one typically requires a powerful computer.

Within this paper, we will focus on the geometric material removal simulation based on voxels. For more details on the complete setup, we would like to refer to [31, 43].

### 2.1 Basic idea of material removal simulation

For the simulation of the process forces, an accurate and detailed description of the tool engagement, instantaneous chip thicknesses, and kinematic cutting conditions have to be determined. For this type of simulation, an in-process description of the changing work piece geometry due to the subtractive machining process is needed. The volumetric model of the work piece has to be able to represent work pieces with large volumes with a high spatial accuracy, while enabling efficient modifications of the work piece geometry. The requirements for such a volumetric model are:

- Representation of large work piece volumes ($\geq 10\,\mathrm{m}^3$)
- High spatial resolution ($\leq 0.001\,\mathrm{mm}^3$)
- Isotropic model and resolution properties
- Efficient geometry modifications ($\geq 10^6$ mods/s)
- Low memory footprint ($\leq 1\,\mathrm{GB/m}^2$ surface area)
- Integrity of the volumetric description by design

State-of-the-art tools mostly rely on dexel-based models. However, dexel simulations have a strong anisotropy in the description of the work piece volume. The dexels have a high resolution along the represented dexel, but a low resolution in between dexels. The number of dexels are also typically limited to a few thousand dexels in each dimension, which can be insufficient for the description of large work pieces with high spatial resolution.

An alternative to dexel models is voxel models. In particular, hierarchical voxel models have the potential to fulfill all the requirements addressed above. Voxel models have nearly isotropic properties and provide an inherently correct volumetric description of the work piece volume. By introducing an hierarchical structure with on demand resolution refinement, large work pieces can be represented with a high spatial resolution, while maintaining a low memory footprint. In comparison to dexel models, the computational effort for the modification of the geometry is larger due to the amount of individual voxel interactions. This disadvantage can be overcome by designing the data structures and modification algorithms for parallel execution on GPUs.

### 2.2 Representation of the work piece

Due to the specific requirements for machining simulation, we will be using a modified octree approach.[2] In order to balance the memory consumption between the actual volumetric information and the inter-layer references between parent and child nodes, the subdivision rate is increased in comparison to the octree structure. Our implementation is inspired by the CPU implementation of the voxel model [31] and the GVDB-library [28] (see Fig. 3). In particular, the dynamic subdivision of the volumetric model and potential modification of the tree structure address efficiently dynamic volumetric structures, as required by the dynamically changing geometry of the work piece.

The smallest spatial unit is a `Voxel`, which represents a cubic section of material with a lateral length of 100 µm and is represented by a single bit—a 1 indicates existing volume, whereas 0 denotes removed material. `Voxels` are organized in containers called `Bricks` as illustrated in blue in Fig. 3. Each `Brick` has a unique id and contains $8^3$, $16^3$, or $32^3$ `Voxel`, addressable by a three dimensional index that directly corresponds to its position in three dimensional space. `Voxels` are either stored in 32- or 64-bit primitive types that allow for atomic operations on the GPU and are laid out linearly within a `Brick`. All `Bricks` are on the lowest level within the hierarchy of the voxel tree and are contained in a structure called `Atlas`.

If we go higher up the hierarchy of the voxel-tree the data-structure is organized by `Nodes`. Each `Node` represents a cube of material equally subdivided by $8^3$ children which are either `Nodes` on a lower level or `Bricks`

[2] The octree data structure splits each cube into 8 smaller ones with 1/2 edge length compared to the original one. The octree data structure is well suited for static objects or slowly/locally changing object structures with memory intensive leaf nodes.

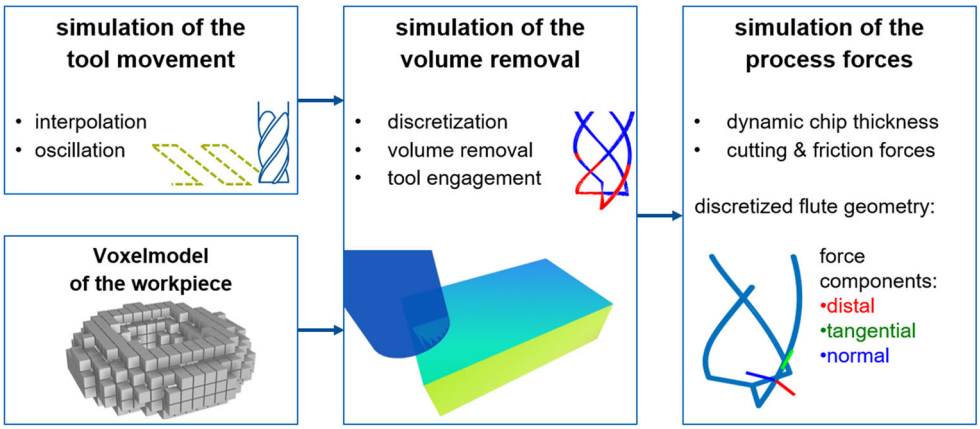


**Fig. 2** Workflow of the process force simulation

on the lowest level, stored as references within a child list. The position of a `Node` is derived from its index in the child list of the parent `Node`. `Nodes` are denoted as active if at least one of their children is active. A `Node` is set as deleted as soon as all of its children are deleted. All `Nodes` of the same hierarchy within the voxel tree have an unique `id` and are part of the same `Level` as depicted in Fig. 3. Several of these `Levels` can be stacked, with its `Nodes` being children of the `Nodes` within the next higher level and parents to the nodes of the child-`Level`. In particular, the children of the lowest level are `Bricks` and the parent of the highest level is called `Root-Node`. The `Root-Node` is a single `Node` that spans the entire available space and serves as entry point for tree traversal. The entirety of all `Levels`, the `Root-Node`, and `Atlas` together is called the `Topology` and resides completely on GPU memory.

The inherent nature of the application implies a constant change of the work piece and thus, `Nodes`/`Bricks` have to be allocated dynamically. This is particularly challenging on the GPU and therefore, the total available number of `Nodes`/`Bricks` within each `Level` and the `Atlas` is predefined and allocated upon initialization. Moreover, a

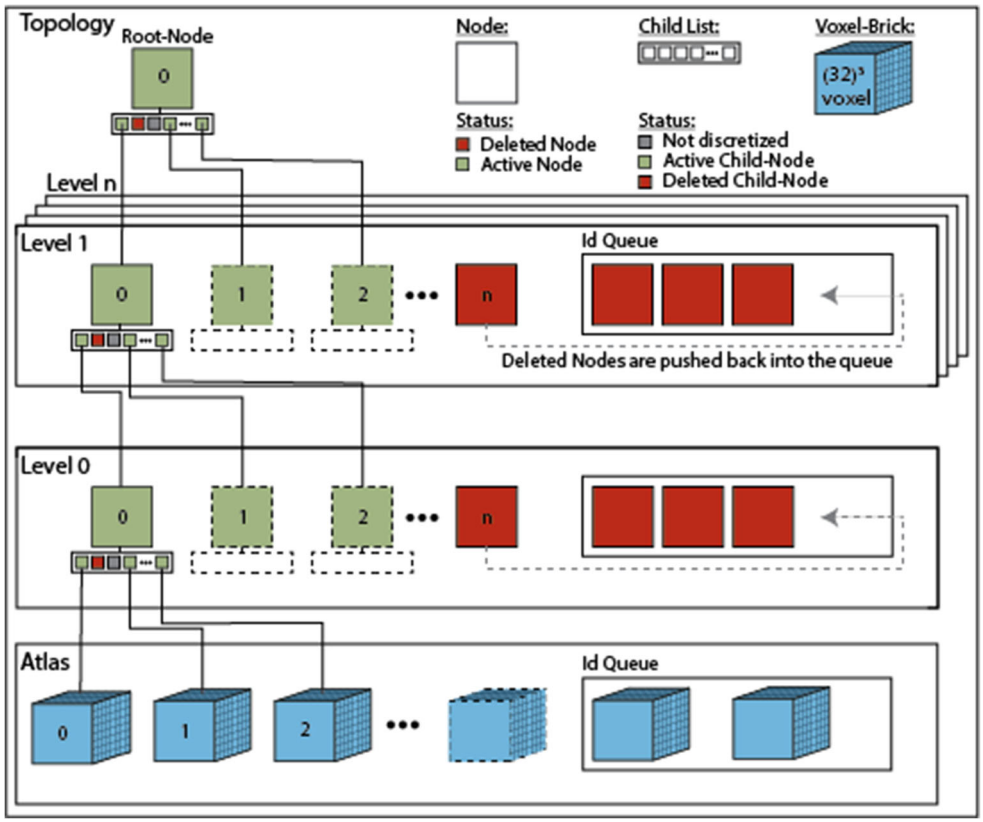


**Fig. 3** Data structure of the voxel tree organized in hierachical levels

queuing system was implemented to dynamically reassign removed Nodes/Bricks within each Level. All indices of unassigned Nodes reside in a queue and are assigned if needed. In case a Node is deleted, its id is pushed to the queue. This way a fast and memory efficient way was used to discretize new regions of the work piece.

### 2.3 Representation of the tool

The geometry of the tool is discretized by equidistant points along the revolute profile of the tool surface. These discrete points interact with the volumetric work piece model and correspond to one entry within one timestep of the engagement histogram. Additional points are located within a surface shell of the tool to ensure the complete removal of coinciding material (see Fig. 4). The coordinates of each point reside on GPU memory with respect to the coordinate system of the tool. The minimum distance $d_{\text{min}}$ between two voxels is given by the orthogonal spatial voxel resolution $r_{\text{spatial}}$. The maximum distance is given by $d_{\text{max}} = \sqrt{3} \cdot r_{\text{spatial}}$. In order to ensure gapless voxel removal and addition, the maximum spatial distance of the discretized tool points $d_{\text{tool}}$ must not exceed the orthogonal spatial voxel resolution $r_{\text{spatial}}$.

### 2.4 Workflow of the simulation

Initially, a Topology is created that is sufficiently large to cover the dimensions of the work piece, which is governed by the number of Levels. Each additional Level extends the lateral length by a factor of 8. The initial volume of the work piece is then voxelized. Nodes, that intersect the surface volume of the work piece, are set active. Those, that entirely reside within the work piece, are set to fully occupied leaf nodes called Volume Nodes, whereas those that lie outside are deleted.

Subsequently, the voxel removal simulation is performed according to the toolpath. The tool path is interpolated with a predefined resolution of 0.07 mm and the engagement between the tool and the work piece is simulated for each step with a given tool transformation matrix as an input and the engagement histogram as an output. The engagement of the tool with the work piece cannot be calculated at once for each tool point individually on the GPU, since this would lead to race conditions. Instead, the engagement process is split up into six consecutive steps (see Fig. 4) in order to allow for parallelization and high performance:

1. Transformation of the discretized tool
2. Mapping the discretized tool points to the topology
3. Updating of the work piece topology
4. Collision detection between tool and work piece
5. Deletion of the engaged voxels
6. Cleanup of the work piece topology

The first step includes the transformation of the discretized tool according to the current transformation along the toolpath and the mapping of the transformed coordinated of the discretized tool points to the according voxel coordinates. This process is performed on the GPU in parallel. The mapped tool coordinates are returned (see Fig. 4).

The second step is called topology mapping and is performed to identify undiscretized regions that are being engaged by the tool. A tree traversal is performed for each tool point and the positions of unassigned Nodes are written to a level specific buffer. These buffers mostly contain duplicates since many tool points coincide with the same unassigned Node. Unique positions are identified using a sorting and consecutive selection step. As a result all positions of unassigned Nodes and Bricks that are going to be engaged by the tool are identified for each Level and the Atlas.

Subsequently, the Topology is updated by configuring all Nodes and Bricks identified in the former step. Starting from the Root-Node, all Nodes are subdivided by allocating child nodes according to the identified engagement positions. At last, new Bricks are configured.

In the following step, the Topology is engaged with the discretized tool points to determine the engagement histogram. A tree traversal is performed for each tool

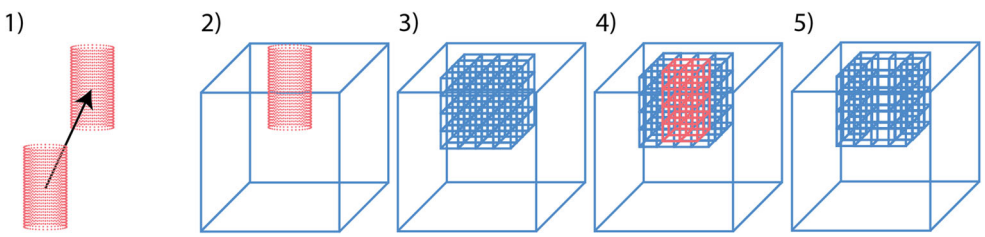


**Fig. 4** Workflow of the simulation: Tool transformation; Mapping of points to the topology; Updating the topology; Engaging the work piece; Removing engaged voxel; Cleaning the topology from empty nodes

point. If the position of a tool point coincides with an existing `Voxel`, the corresponding entry in the engagement histogram is set to one.

Consecutively, every `Voxel` that has interacted with the tool has to be deleted, which is achieved in a separate computation step using atomic and-operations to make sure that only one bit is changed at the same time.

The `Topology` is then cleaned to free unnecessary occupied memory by empty `Nodes` and `Atlases`. This procedure is executed starting from the `Atlas` level and all the way up to the `Root-Node`. This is initially done by summing up the `Voxels` within each active `Brick`. If the respective sum equals to zero, the `brick id` is appended to the `Atlas`'s `id` queue and its entry within its parent's `Node` child list is deleted. A similar process is performed for each additional `Level`. The number of deleted children is summed up for each active `Node` and it is appended to the respective `id` queue if all of its children are deleted. The described cleanup procedure is executed once for every 1000 engagement operations in order to increase the performance of the overall material removal simulation.

After all the described steps are performed the engagement histogram (Fig. 5), i.e. which points of the tool geometry (Section 2.3) interacted with the material (Section 2.2), is copied to the CPU to prepare for the process force calculation.

## 2.5 Process force simulation

In the first step, the resulting engagement histograms of the volume removal simulation step are temporally interpolated for the determination of the engagement status of a discretized cutting edge point of the tool with the work piece. The intersecting finite cutting edge segments are given by the subset $\mathcal{C} \in \mathcal{F}$, where $\mathcal{F}$ represent all discretized cutting edge points. Each discretized cutting edge point is represented by a cutting edge aligned coordinate system $\mathcal{K}_i$. The local coordinate system $\mathcal{K}_i$ is defined by the position of the discretized cutting edge point $\boldsymbol{r}_i$, a tangential component $\boldsymbol{t}_i$ along the local cutting edge tangent, a distal component $\boldsymbol{d}_i$ pointing in outwards direction and a normal component $\boldsymbol{n}_i$ perpendicular to the tangential and distal component pointing towards the cutting velocity direction. The discretized tool geometry is shown in Fig. 6.

The local uncut chip thickness $h_i$ is calculated based on the spindle speed $n$, the number of flutes $z$ and the translational and rotational velocity $\boldsymbol{v}_{\text{trans}}$ and $\boldsymbol{v}_{\text{rot}}$ of the tool centre point (see Fig. 6). The velocities include the ideal tool movements and the tool movements resulting from dynamic oscillations of the machine structure. The temporal dependency of the required values involved in the simulation of the cutting forces is omitted in the description of the mathematical notation for better readability.

$$\text{flute segment length:} \quad s_l = |\boldsymbol{r}_{i+1} - \boldsymbol{r}_i| \tag{1}$$

$$\text{tangential component:} \quad \boldsymbol{t}_i = (\boldsymbol{r}_{i+1} - \boldsymbol{r}_i)/s_l \tag{2}$$

$$\text{cutting direction:} \quad \boldsymbol{v}_{c,i} = (\boldsymbol{r}_i \times \boldsymbol{z})/|\boldsymbol{r}_i \times \boldsymbol{z}| \tag{3}$$

$$\text{distal component:} \quad \boldsymbol{d}_i = (\boldsymbol{r}_i \times \boldsymbol{z})/|\boldsymbol{r}_i \times \boldsymbol{z}| \tag{4}$$

$$\text{normal component:} \quad \boldsymbol{n}_i = (\boldsymbol{t}_i \times \boldsymbol{v}_{c,i})/|\boldsymbol{t}_i \times \boldsymbol{v}_{c,i}| \tag{5}$$

$$\text{periodic flute time:} \quad dt = 1.0/(n \cdot z) \tag{6}$$

$$\text{chip thickness:} \quad h_i = \boldsymbol{d}_i \cdot (\boldsymbol{v}_{\text{trans}} + \boldsymbol{v}_{\text{rot}} \times \boldsymbol{r}_i) \cdot dt \tag{7}$$

The instantaneous chip thickness $h_i$ is used for the mechanistic computation of the instantaneous cutting forces $\boldsymbol{F}$. The cutting force parameters for the tangential, distal and normal friction and cutting components are stated by $k_{te}$,

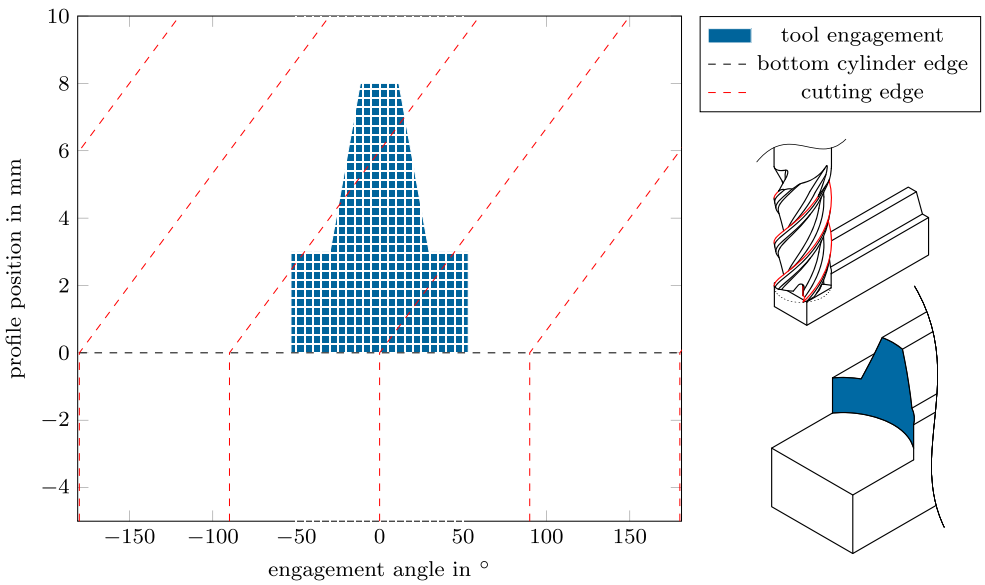


**Fig. 5** Schematic representation of the generation of engagement histograms

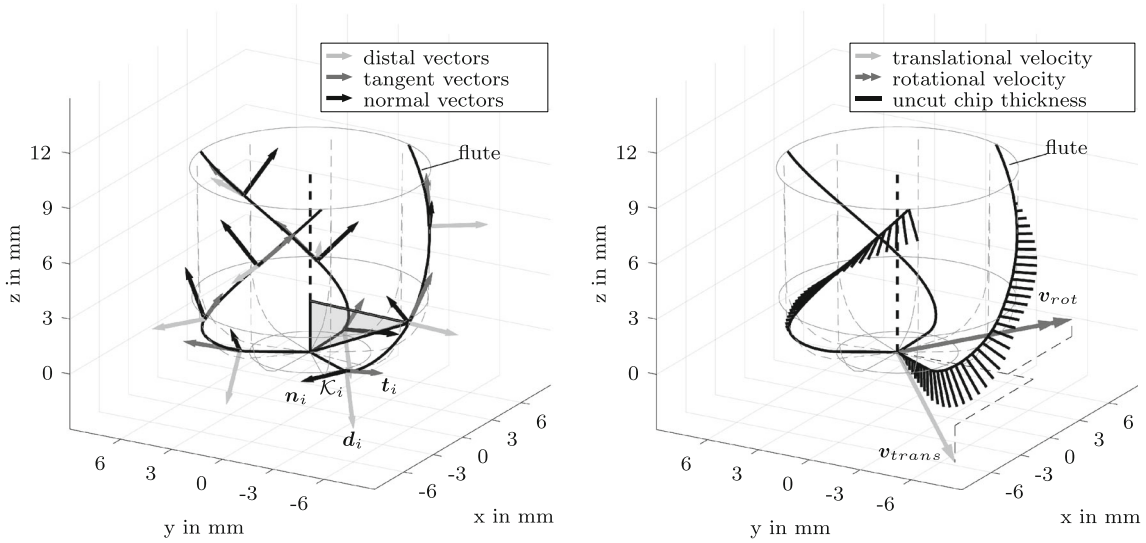


**Fig. 6** Spatial representation of the discretized tool geometry and the simulated chip thickness [31]

$k_{tc}$, $k_{de}$, $k_{dc}$, $k_{ne}$ and $k_{nc}$.

$$\boldsymbol{F} = \sum_{i \in \mathcal{C}} \begin{bmatrix} \boldsymbol{t}_i & \boldsymbol{d}_i & \boldsymbol{n}_i \end{bmatrix} \begin{bmatrix} k_{te} + k_{tc} \cdot h_i \\ k_{de} + k_{dc} \cdot h_i \\ k_{ne} + k_{nc} \cdot h_i \end{bmatrix} \tag{8}$$

A detailed description of the process force simulation, a schematic visualization of the tool geometry and a validation of the process force model are provided in [31].

## 3 Performance and accuracy evaluation

Within this section, we will benchmark the proposed approach along six benchmark work pieces of different size, shape, and machining operations (Section 3.1). To do so, we compare the time required to simulate material removal for a CPU and a GPU implementation of the voxel model on different computers (Section 3.2) as well as compare the quality of a voxel model with a dexel model (Section 3.3).

### 3.1 Test parts and computational resources

The considered benchmark work pieces are shown in Fig. 7. The features of the work pieces are described in more detail below. The simulation of the milling process covers three different tools with varying numbers of discretization points along the tool profile and constant angular discretization steps for the generation of the revolute shell. The tool length, the tool diameter, the number of discretized shell points for the determination of the tool engagement and the number of discretized interior tool geometry points for gapless volume removal are given in Table 2.

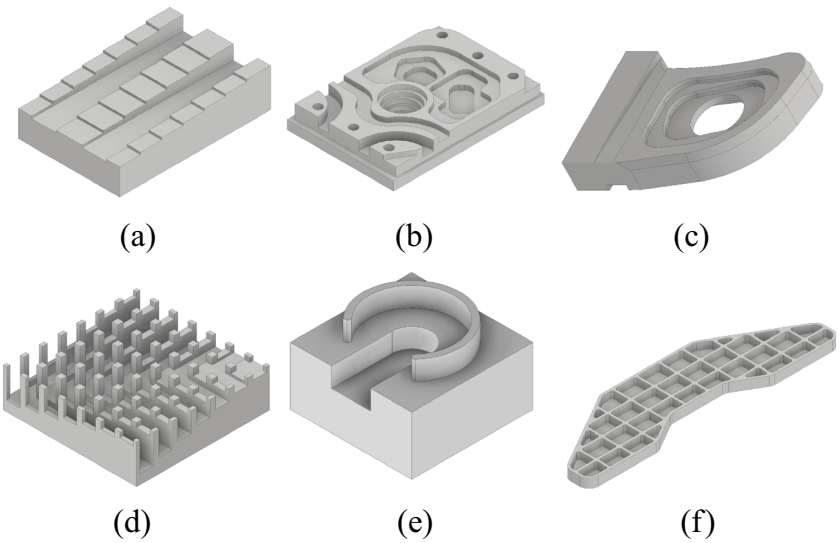


**Fig. 7** Test cases/benchmark parts: **a** simple test part; **b** complex test part; **c** freeform part; **d** radiator part; **e** roughing part; **f** lightweight part

**Table 2** Different tools used for the milling simulation process

| | D12 CR0 L7 HA45 | D12 CR0 L10 HA45 | D12 CR0 L27 HA45 | D30 CR0 L16 HA45 |
|---|---|---|---|---|
| Length | 7.0 mm | 10.0 mm | 27.0 mm | 16.0 mm |
| Diameter | 12.0 mm | 12.0 mm | 12.0 mm | 30.0 mm |
| Shell points | 46,800 | 57,600 | 118,800 | 111,600 |
| vol. points | 265,856 | 345,936 | 798,388 | 1,569,374 |

Three different computational resources are used to validate the different algorithms on different hardware setups (see Table 3) and also to compare their performance with respect to different GPUs.

**Example (a)—Simple Test Part** As the basic validation we have chosen a simple linear cut with increasing depth of cut along the tool path. Despite the simple geometry of the part, the engagement of the tool and the simulated process forces change within and in between the steps.

**Example (b)—Complex Test Part** To increase the complexity we have chosen a more complex part with different features as they appear in typical machining tasks. The complex test part includes full engagement cuts with changing directions, outer profiles and pockets. This part is also used for validation in real experiments (see Section 4).

**Example (c)—Freeform Part** To further increase complexity, we consider a test part including free form surfaces, where the tool angle ($z$-axis) is changing. This results in 5-axis machining processes. This part is also referenced in the comparison of the geometric accuracy of the dexel and voxel models (see Section 3.3).

**Example (d)—Radiator Part** To investigate the behaviour of the methods for parts with a large surface to volume ratio, we have selected a radiator like structure with additional complexity added. Furthermore, the example addresses intersections of already removed material areas. A crucial test for correct allocation of memory.

**Example (e)—Roughing Part** All features of the part are machined with a large diameter tool resulting in tool selections and tool path comparable to roughing operations. Furthermore, we simulate the same part with a smaller tool to compare the effect of tool sizes on the performance of the presented voxel model.

**Table 3** Hardware setups used in the computational studies; GPU2 is an Amazon EC2 P3.2x large instance

| Computer CPU1 | Computer GPU1 | Computer GPU2 |
|---|---|---|
| Core i7-6700k | Core i7-8750H | 8 Core CPU |
| 32GB RAM | 4GB VRAM | 16GB VRAM |
| GTX 1070 | Quadro P1000 | Tesla V100 |

**Example (f)—Aerospace Part** Last but not least, we benchmark the part along a large and complex part as it might occur in the aerospace industry. Aerospace parts are often characterized by thin-walled parts with large outer dimensions and a buy to fly ratio of over 95%. The simulation of the material removal process for these parts typically leads to trade-offs between memory consumption, simulation time and spatial resolution, which can impact the simulation accuracy. Being one of applications with high potential for robotic milling, the proposed voxel model has to be able to accurately simulate the material removal process without compromises in simulation speed and spatial accuracy.

### 3.2 Computational experiments

Table 4 shows a performance comparison of the voxel model for the geometries shown in Fig. 7 on different computer architectures. The initial stock sizes are determined by the axis aligned bounding boxes of the part geometries and range from 17 to 13, 200 $cm^3$. 25–43% of the material is removed during the simulated manufacturing process.

Based on the presented voxel model, the simulated material removal process was successfully executed for all part geometries shown in Fig. 7, where the freeform part was used to test 5-axis simultaneous machining processes and the lightweight part was used to test the memory consumption. On the high performance GPU2, the largest part with a volume of 13,200 $cm^3$ and a 43% material removal rate takes less than 60 s to compute. Compared to the estimated manufacturing time of 2 h 20 min, the simulation speed exceeds the manufacturing speed by a factor of 140. The GPU utilization during the material removal simulation was typically larger than 70%. In many of the test cases, a GPU load of over 90% was achieved, indicating a good parallelization and a high efficiency of the simulation process. While not being directly comparable, the simulation of the manufacturing of a partial impeller

**Table 4** Comparison of paths (mm/segments), timings (s), computational efficiency (% of GPU utilization, i.e. percentage of time the GPU is actively used) for different parts, tools, and computers

| | Simple part | Complex part | Freeform part | Radiator part | Lightweight part |
|---|---|---|---|---|---|
| Length | 75 mm | 120 mm | 80 mm | 120 mm | 1100 mm |
| Width | 50 mm | 170 mm | 20 mm | 120 mm | 300 mm |
| Height | 19 mm | 25 mm | 120 mm | 35 mm | 40 mm |
| Tool | D12 CR0 L10 HA45 | D12 CR0 L10 HA45 | D12 CR0 L7 HA45 | D12 CR0 L27 HA45 | D12 CR0 L27 HA45 |
| Path | 1676 mm | 13,914 mm | 13,053 mm | 2911 mm | 74,340 mm |
| | 22,500 seg | 149,748 seg | 174,674 seg | 45,732 seg | 846,111 seg |
| Removed volume | 18.8 $cm^3$ | 127.0 $cm^3$ | 50.7 $cm^3$ | 218.5 $cm^3$ | 5548.2 $cm^3$ |
| Voxel CPU1 | 0.40 s | 86.41 s | 5.87 s | 35.72 s | 1583.5 s |
| Voxel GPU1 | 8.02 s | 55.5 s | 41.57 s | 39.91 s | 675.60 s |
| | 70.01% | 89.9% | 89.2% | 73.8% | 76.2% |
| Voxel GPU2 | 0.61 s | 4.41 s | 3.11 s | 2.90 s | 59.94 s |
| | 80.7% | 75.3% | 67.7% | 93.7% | 91.7% |

with 1024 dexel subdivisions, which would result in a spatial dexel grid resolution of approximately 1 mm for the lightweight part, requires 78 min [23].

For the smaller parts, e.g. the Simple Test Part, the GPU implementation performs slower than the CPU implementation of the voxel model. Additional overheads for the initiation and execution of the memory transfer limit the simulation speed for small tools and parts. The overall simulation time for small parts lies within the range of seconds and does not pose a problem for the typical user. Comparing the performance of the GPU1 and GPU2 we can observe for all cases a speedup factor of 10–13, which corresponds roughly to the difference in floating point performance of the two GPUs (1894 gflops for GPU1 vs 14,131 gflops for GPU2).

Last but not least, we have compared the effect of the machining tool (Table 5) on computing times. To do so, we have simulated the roughing part (Fig. 7e) with two different tool geometries as shown in Fig. 5. For both, CPU and GPU, the run time is about half time with small tool than with big tool. Nevertheless the speed-up factor for big tool from CPU to GPU is around 25, for small tool around 18. This is due to the better capacity utilization of GPU with increasing number of tool-points.

The benchmark tests have shown significant performance increases of the parallel GPU implementation and a good scaling of the performance with the number of available computational units, the work piece and the tool size.

**Table 5** Comparison of timings for the roughing part using two different tools

| | Big tool | Small tool |
|---|---|---|
| Length | 100 mm | 100 mm |
| Width | 100 mm | 100 mm |
| Height | 64 mm | 64 mm |
| Tool | D30 CR0 L16 HA45 | D12 CR0 L10 HA45 |
| Path | 2028 mm | 2794 mm |
| | 21,346 seg | 31,913 seg |
| Removed volume | 179.6 $cm^3$ | 179.6 $cm^3$ |
| Voxel CPU1 | 62.54 s | 34.61 s |
| Voxel GPU1 | 33.99 s | 18.80 s |
| | 81.6% | 81.3% |
| Voxel GPU2 | 2.55 s | 1.82 s |
| | 96.5% | 93.2% |

### 3.3 Accuracy of voxel and dexel models

Due to the closed source solution of commercial state-of-the-art dexel simulation environments (e.g. [34]), the simulation time cannot be analysed in detail. The resulting geometry of the reconstructed finished part on the other hand can be exported and compared to the reconstructed part geometry of the proposed voxel model.

Due to the an-isotropic properties of dexel models, the reconstruction of the work piece geometry can suffer from aliasing, artefacts and inaccuracies [1]. These effects reduce the accuracy of the tool engagement and chip thickness reconstruction. Since the process force calculation is based on the simulated uncut chip geometry, the prediction of process forces is affected.

While axis aligned and flat surfaces are typically reconstructed with high precision, curved and twofold surfaces, small features and sharp edges pose challenges to the reconstruction of the work piece volume using dexel based approaches. The reconstruction accuracy of the work piece surface relies on the direction of the dexels and is thus highly an-isotropic. While voxel models still exhibiting an-isotropic properties, these effects are limited by the

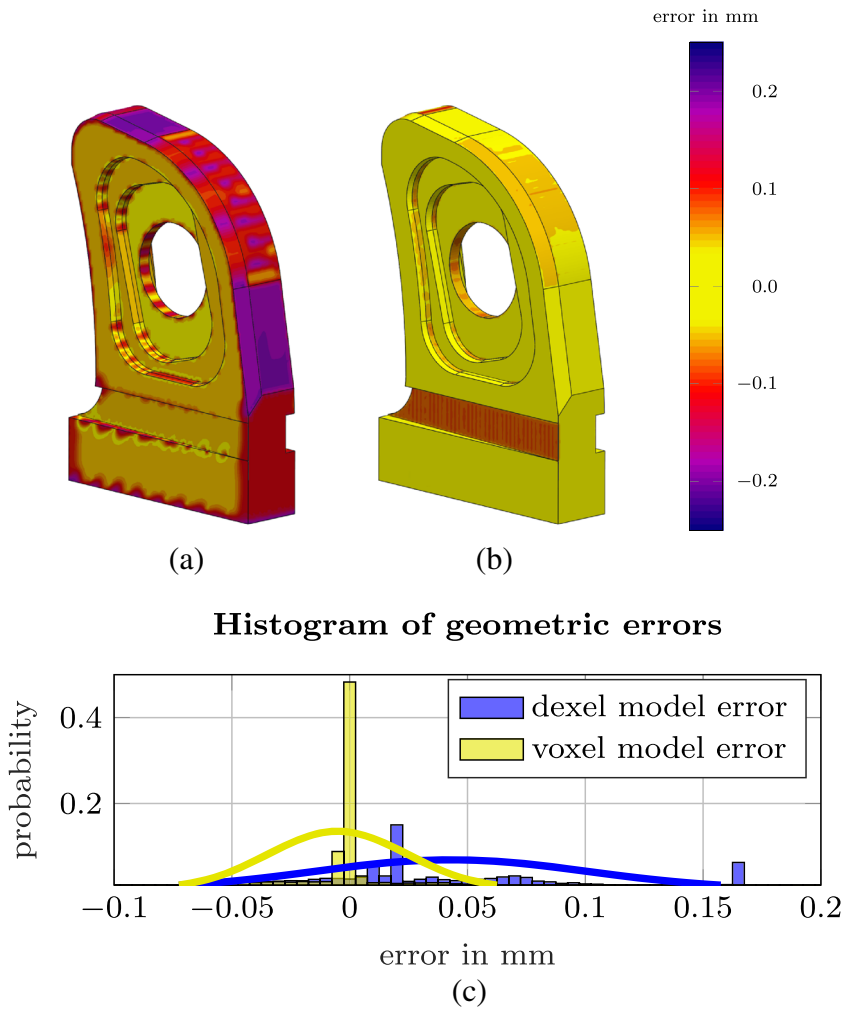


**Fig. 8** Comparison of the form error of a commercial implementation of a dexel model for CAM machining simulation (**a**) and the presented voxel model implementation (**b**). Error histogram and fitted normal error distribution for the dexel and voxel models (**c**)

geometry of the voxel. Thereby, the spatial resolution of the voxel model varies between the length of the edge of one voxel cube and the length of the diagonal of one voxel cube depending on the sampling direction. For the volumetric sampling being executed at the centre of the voxel, the expected surface error lies within half of the edge and diagonal length.

The spatial accuracy of the proposed voxel model is benchmarked against a commercial implementation of a dexel work piece model [34] as typically found in machining simulation environments. For the comparison, the machining of the freeform part (see Fig. 7c) is simulated. The resulting surface representation is exported for both the dexel and the voxel models and compared to the ideal CAD geometry. Figure 8 shows the color coded surface geometry error of the dexel and voxel models.

Table 6 shows the statistical analysis of the surface geometry error of the dexel and the voxel models.

The color coded comparison (see Fig. 8) and the statistical analysis (see Table 6) both show a higher accuracy of the voxel model compared to the dexel model, while the characteristics of the form error differ. The form errors of the reconstructed dexel model show form errors due to local approximation by flat surfaces, while the voxel model shows spatial aliasing. Both models show a directional dependency of the surface error, while being more prominent for the dexel model. Commercial dexel model implementations typically subdivide the outer

**Table 6** Comparison of the form error of a commercial dexel model and the proposed voxel model

| Model error in mm | MIN | MAX | MEAN | RMS | STD |
|---|---|---|---|---|---|
| Dexel | − 0.2435 | 0.2297 | 0.043 | 0.0721 | 0.0579 |
| Voxel | − 0.134 | 0.0431 | − 0.0241 | 0.0448 | 0.0377 |

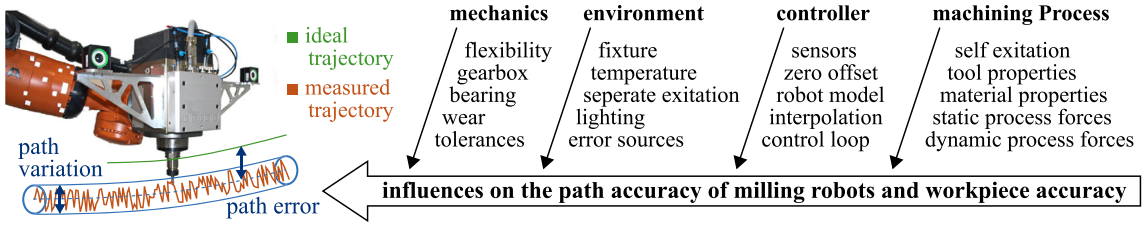


**Fig. 9** Influences on the path accuracy of milling robots, based on [33] (p. 1403)

dimensions of the work piece with a predefined fixed or limited number of individual dexels. Therefore, the error of the commercially implemented dexel models typically depends on the outer dimensions of the part. The spatial resolution of the presented voxel model is invariant to the size of the work piece. The presented reconstruction errors for the Freeform part with small outer dimensions are therefore in favor for the dexel model. The reconstruction errors of the dexel models can be significantly larger for parts with larger outer dimensions.

## 4 Robot milling

Robot-based machining systems have the potential to offer a cost-effective alternative to conventional machine tools. The combination of a large available workspace and a dexterous kinematic configuration facilitates the machining of large work pieces with complex geometry and undercuts. In comparison to conventional machine tools, robotic systems are optimized for rapid movements, repetitive position accuracy and light-weight of the structural components. Therefore, robots have lower eigenmodes, static stiffness and path accuracy. The dominant influences on the path accuracy of machining robots are listed in Fig. 9.

Milling combines challenging demands on the system behaviour of the robot with additional external process forces. While standard pick-and-place robot tasks can be programmed without the consideration of the process, the planning of machining operations for milling robots have to take coupled machine-process-interaction into consideration. The presented volumetric work piece model is the core component of the coupled simulation of the system behaviour.

While the inherent static and dynamic forces of the robot system can be calculated and measured based on the robot movement and the joint torques, unknown external forces pose a challenge to the robot controller and cannot be compensated without additional sensors. The external forces cause an elastic deformation of the gears and bearings, which cannot be measured by the encoders mounted on the motor side. By simulating the process forces and taking them into account when generating and executing the part programs, the resulting path deviations can be compensated. A detailed description of the simulation of the process forces and the generation of the part programs can be found in [43].

For the validation of the simulation-enhanced robot-based machining process, the manufacturing of the complex cam work piece (see Fig. 7b) was planned, simulated and machined utilizing the presented method. For the simulation of the machining of aluminium EN AW-2007 the process force parameters are given in Table 7 [31]. Figure 10 shows a comparison between the measured and simulated process forces for two exemplary sections during the manufacturing process.

By coupling the process force simulation and the dynamic flexible model of the robotic system, the force-induced deflections during the machining process can be simulated and compensated. A feed-forward controller converts the simulated forces to expected path deviations and superimposes the mirrored path errors to the original tool path. Figure 11 shows the resulting geometry errors without and with activated compensation mechanism.

The voxel-based process force simulation and compensation have shown a reduction of the remaining RMS contour error by 64%. Thereby, 99% of the remaining errors have been smaller than 0.25 mm, which enables the utilization of milling robots for roughing operations. Furthermore, the presented voxel-based approach allows for the representation of large work pieces without impairing the spatial accuracy of the in-process work piece. This property allows for the simulation of the machining of large work pieces typically manufactured by robotic machining systems.

## 5 Discussion

In this article, we have introduced a novel concept for fast and accurate process force prediction of machining. The

**Table 7** Examplary process force parameters for EN AW-2007

| | |
|---|---|
| $k_{te} = 4.63 \frac{N}{mm}$ | $k_{tc} = 57.5 \frac{N}{mm^2}$ |
| $k_{de} = -11.8 \frac{N}{mm}$ | $k_{dc} = -76.5 \frac{N}{mm^2}$ |
| $k_{ne} = -11.0 \frac{N}{mm}$ | $k_{nc} = -672 \frac{N}{mm^2}$ |

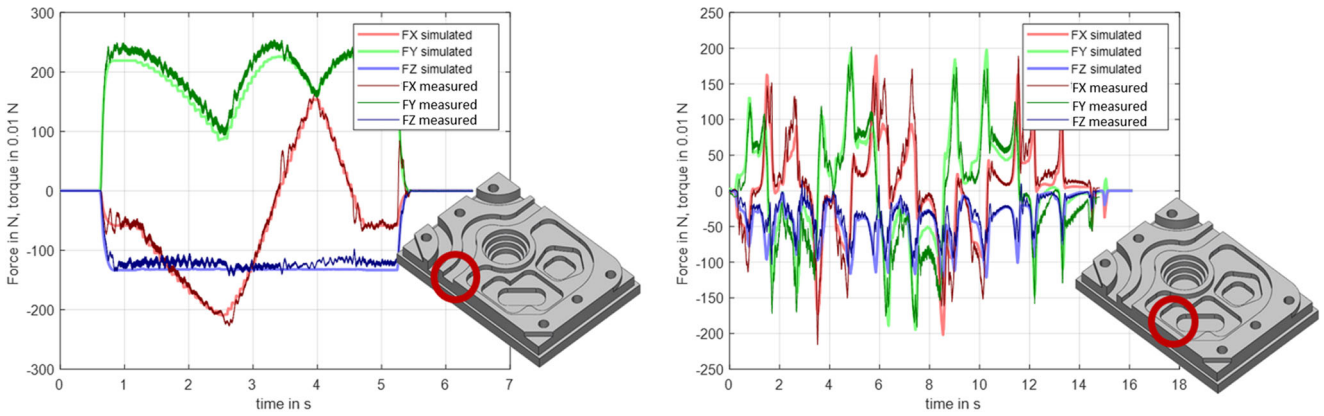


**Fig. 10** Comparison of the simulated and measured process forces for the machining of a slot with full tool engagement and changing directions of cut (**a**) and the machining of a pocket (**b**)

approach is based on the concept introduced by [31], which splits the force calculation in two parts: a geometric material removal simulation (Sections 2.2–2.4) and a mechanistic process force simulation (Section 2.5). This allows, e.g., to perform the material removal simulation upfront on a standard desktop computer and the force evaluation can be realized on an edge device online to the process itself.

Within this article, we focused on the acceleration on the computationally demanding geometry material removal simulation. While many state-of-the-art approaches rely on dexel realizations we rely on hierarchical voxel models. This does not only allow for efficient GPU implementations as shown in Section 3.2 but also lead to more accurate shape representations as shown in Section 3.3.

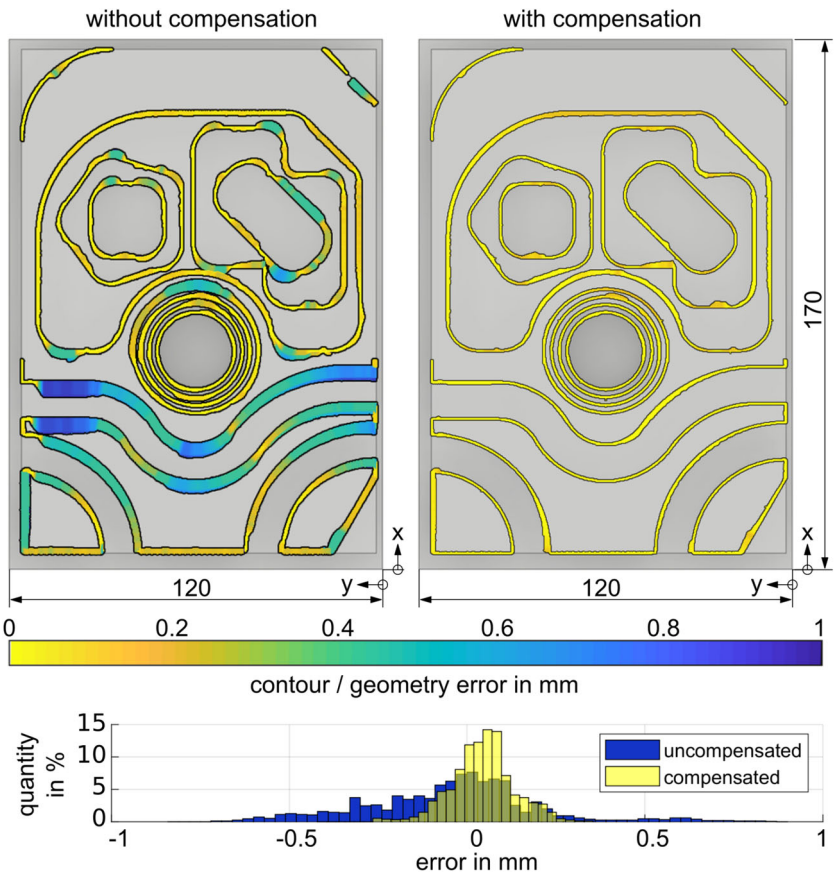


**Fig. 11** Visualization of the contour error without (top left) and with (top right) static process force compensation and a histogram of the remaining contour errors (bottom)

By utilizing the presented voxel model for the improvement of robot milling applications, the achievable roughing accuracy was improved from the range of 1.0 mm to the range of 0.1–0.2 mm. Besides this use case, the presented model could be beneficial for a large number of other applications:

- Increased accuracy for low stiffness machine tools
- Process parameter optimization in machining
- Simulated feedback during the CAM-planning
- Machine tool and spindle selection
- Estimation of manufacturing costs
- Machine tool monitoring, diagnostics and predictive maintenance

The presented voxel-based approach is able to represent large mutable volumes with high mutation performance, nearly isotropic properties, inherent model integrity and high spatial resolution. These properties of the presented model are achieved by a combination of a voxelization approach, hierarchical dynamic data structures, and the parallel implementation of query and mutation operations on the GPU. Fulfilling the requirements for various simulation applications in the area of virtual machining, additive manufacturing, surgical simulations, or excavation simulation, we expect to see the base technology to be employed by other researchers and developers.

**Funding** Open Access funding enabled and organized by Projekt DEAL. The research was largely funded by Siemens Technology.

**Data availability** Data and material exceeding the information within this publication cannot be published at the moment.

**Code availability** Releasing the code under an open-source license is under active discussion.

## Declarations

**Conflict of interest** The authors declare no competing interests.



## References

1. Altintas Y, Kersting P, Biermann D, Budak E, Denkena B, Lazoglu I (2014) Virtual process systems for part machining operations. CIRP Ann 63(2):585–605
2. Armendia M, Ghassempouri M, Ozturk E, Peysson F (2019) Twin-control: a digital twin approach to improve machine tools lifecycle. Springer Nature, Berlin
3. Boess V, Ammermann C, Niederwestberg D, Denkena B (2012) Contact zone analysis based on multidexel workpiece model and detailed tool geometry representation. Procedia CIRP 4:41–45
4. Böß V, Denkena B, Breidenstein B, Dittrich MA, Nguyen HN (2019) Improving technological machining simulation by tailored workpiece models and kinematics. Procedia CIRP 82:224–230
5. Bouhadja K, Bey M (2014) Classification of simulation methods in machining on multi-axis machines. In: Proceedings of the world congress on engineering, vol 2. pp 992–997
6. Bouhadja K, Bey M (2015) Survey on simulation methods in multi-axis machining. In: Transactions on engineering technologies. Springer, pp 367–382
7. Cao X, Zhao G, Xiao W (2020) Digital twin–oriented real-time cutting simulation for intelligent computer numerical control machining. In: Proceedings of the institution of mechanical engineers part b: journal of engineering manufacture
8. Chen Y, Dong F (2013) Robot machining: recent development and future research issues. Int J Adv Manuf Technol 66(9-12):1489–1497
9. Cheng H, Gao J, Kafka OL, Zhang K, Luo B, Liu WK (2017) A micro-scale cutting model for ud cfrp composites with thermo-mechanical coupling. Compos Sci Technol 153:18–31
10. Cordes M, Hintze W (2017) Offline simulation of path deviation due to joint compliance and hysteresis for robot machining. Int J Adv Manuf Technol 90(1-4):1075–1083
11. Denkena B, Böß V., Nespor D, Rust F (2015) Simulation and evaluation of different process strategies in a 5-axis re-contouring process. Procedia Cirp 35:31–37
12. Denkena B, Grove T, Pape O (2019) Optimization of complex cutting tools using a multi-dexel based material removal simulation. Procedia CIRP 82:379–382
13. Doubrovski EL, Tsai EY, Dikovsky D, Geraedts JM, Herr H, Oxman N (2015) Voxel-based fabrication through material property mapping: A design method for bitmap printing. Comput Aided Des 60:3–13
14. Elser A, Königs M., Verl A, Servos M (2018) On achieving accuracy and efficiency in additive manufacturing: Requirements on a hybrid cam system. Procedia CIRP 72:1512–1517
15. Engin S, Altintas Y (2001) Mechanics and dynamics of general milling cutters.: Part i: helical end mills. Int J Mach Tools Manuf 41(15):2195–2212
16. Gavranovic S, Hartmann D, Wever U (2019) Topology optimization using gpgpu. In: Advances in evolutionary and deterministic methods for design, optimization and control in engineering and sciences. Springer, pp 553–566
17. Groover M, Zimmers E (1983) CAD/CAM: computer-aided design and manufacturing. Pearson Education, London
18. He S, Zeng X, Yan C, Gong H, Lee CH (2017) Tri-dexel model based geometric simulation of multi-axis additive manufacturing. In: International conference on intelligent robotics and applications. Springer, pp 819–830
19. Inui M, Huang Y, Onozuka H, Umezu N (2020) Geometric simulation of power skiving of internal gear using solid model with triple-dexel representation. Procedia Manuf 48:520–527

20. Inui M, Kaneda M, Kakio R (1999) Fast simulation of sculptured surface milling with 3-axis nc machine. In: Machining impossible shapes.Springer, pp 97–108
21. Jang D, Kim K, Jung J (2000) Voxel-based virtual multi-axis machining. Int J Adv Manuf Technol 16(10):709–713
22. Ji W, Wang L (2019) Industrial robotic machining: a review. Int J Adv Manuf Technol 103(1-4):1239–1255
23. Lee SW, Nestler A (2012) Virtual workpiece: workpiece representation for material removal process. Int J Adv Manuf Technol 58(5):443–463
24. McCloskey P (2019) Virtual model of power skiving cutting mechanics. Ph.D. thesis, University of Waterloo
25. ModuleWorks: Machine Simulation. https://www.moduleworks.com/software-components/simulation/%#machine-simulation
26. Museth K, Lait J, Johanson J, Budsberg J, Henderson R, Alden M, Cucka P, Hill D, Pearce A (2013) Openvdb: an open-source data structure and toolkit for high-resolution volumes. In: Acm siggraph 2013 courses, pp 1–1
27. Peng X, Zhang W (2009) A virtual sculpting system based on triple dexel models with haptics. Comput-Aid Des Appl 6(5):645–659
28. Hoetzlein R GVDB: Raytracing sparse voxel databasestructures on the GPU. https://developer.nvidia.com/gvdb
29. Reinl C, Friedmann M, Bauer J, Pischan M, Abele E, Von Stryk O (2011) Model-based off-line compensation of path deviation for industrial robots in milling applications. In: 2011 IEEE/ASME international conference on advanced intelligent mechatronics (AIM). IEEE, pp 367–372
30. Roscoe L et al (1988) Stereolithography interface specification. Am-3D Syst Inc 27
31. Schnoes F, Zaeh M (2019) Model-based planning of machining operations for industrial robots. Procedia CIRP 82:497–502
32. Sethian J, Sethian J (1996) Level set methods: Evolving interfaces in geometry, fluid mechanics, computer vision, and materials science. Cambridge University Press, Cambridge
33. Siciliano B, Khatib O (2016) Springer handbook of robotics, Springer, Berlin
34. Siemens NX Cam. https://www.plm.automation.siemens.com/global/en/products/nx/nx-for-manufacturing.html
35. Slavkovic NR, Milutinovic DS, Glavonjic MM (2014) A method for off-line compensation of cutting force-induced errors in robotic machining by tool path modification. Int J Adv Manuf Technol 70(9-12):2083–2096
36. Stifter S (1995) Simulation of nc machining based on the dexel model: a critical analysis. Int J Adv Manuf Technol 10(3):149–157
37. Surleraux A, Bigot S, Pernot JP, d'Urso G, Merla C (2015) Using voxels in the simulation of manufacturing processes
38. Tukora B (2012) Material removal simulation and cutting force prediction of multi-axis machining processes on general-purpose graphics processing units
39. Ueng SK, Chen LG, Jen SY (2018) Voxel-based virtual manufacturing simulation for three-dimensional printing. Adv Mech Eng 10(6):1687814018781632
40. Verl A, Valente A, Melkote S, Brecher C, Ozturk E, Tunc LT (2019) Robots in machining. CIRP Ann 68(2):799–822
41. Wang Z, Chen JSS, Joy J, Feng HY (2018) Machined sharp edge restoration for triangle mesh workpiece models derived from grid-based machining simulation. Comput-Aid Des Appl 15(6):905–915
42. Wimmer S, Zaeh M (2018) The prediction of surface error characteristics in the peripheral milling of thin-walled structures. J Manuf Mater Process 2(1):13
43. Zaeh M, Schnoes F, Obst B, Hartmann D (2020) Combined offline simulation and online adaptation approach for the accuracy improvement of milling robots. CIRP Ann
44. Zaeh M, Roesch O (2014) Improvement of the machining accuracy of milling robots. Prod Eng 8(6):737–744

# Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com