Machine Vision and Applications (2021) 32:8
https://doi.org/10.1007/s00138-020-01116-y

ORIGINAL PAPER

Feature-Driven Viewpoint Placement for Model-Based Surface Inspection

Dennis Mosbach[1,2] · Petra Gospodnetić[1,2] · Markus Rauhut[2] · Bernd Hamann[3] · Hans Hagen[1]

Received: 25 July 2019 / Revised: 10 July 2020 / Accepted: 20 August 2020

Abstract
The goal of visual surface inspection is to analyze an object's surface and detect defects by looking at it from different angles. Developments over the past years have made it possible to partially automate this process. Inspection systems use robots to move cameras and obtain pictures that are evaluated by image processing algorithms. Setting up these systems or adapting them to new models is primarily done manually. A key challenge is to define camera viewpoints from which the images are taken. The number of viewpoints should be as low as possible while still guaranteeing an inspection of the desired quality. System engineers define and evaluate configurations that are improved based on a time-consuming trial-and-error process leading to a sufficient, but not necessarily optimal, configuration. With the availability of 3D surface models defined by triangular meshes, this step can be done virtually. This paper presents a new scalable approach to determine a small number of well-placed camera viewpoints for optical surface inspection planning. The initial model is approximated by B-spline surfaces. A set of geometric feature functionals is defined and used for an adaptive, non-uniform surface sampling that is sparse in geometrically low-complexity areas and dense in regions of higher complexity. The presented approach is applicable to solid objects with a given 3D surface model. It makes camera viewpoint generation independent of the resolution of the triangle mesh, and it improves previous results considering number of viewpoints and their relevance.

Keywords View planning · Object space exploration · Inspection automation · B-spline surface · Model-based planning

1 Introduction

With the abundance of various sensing methods, almost every produced object undergoes a quality assurance process. Because of their availability and application versatility, optical sensors are commonly utilized, both as a support and a primary sensing device. When it comes to surface quality inspection, they are especially useful as there are many advanced machine vision tools available (e.g., [3,10,15]). Moreover, they enable the possibility of inspecting both optical and spatial properties of the object. Over the years, significant effort has been put into inspection process automation. An in-depth analysis of the topic and various ways to automate machine vision tasks have already been covered by the literature [2].

However, the automation of visual inspection is inextricably connected to the application and inspection environment; it is therefore necessary that an expert designs the image acquisition system and hardware configuration. The design of the system affects the acquired images and consequently enables or hinders the machine vision algorithms used for analyzing images.

The expert approach is required when a decision concerning the hardware placement is needed. The system design is an iterative process consisting of four steps: *assessment* of

✉ Dennis Mosbach
dennis.mosbach@itwm.fraunhofer.de

Petra Gospodnetić
petra.gospodnetic@itwm.fraunhofer.de

Markus Rauhut
markus.rauhut@itwm.fraunhofer.de

Bernd Hamann
hamann@cs.ucdavis.edu

Hans Hagen
hagen@cs.uni-kl.de

[1] Computer Graphics and HCI Group, University of Kaiserslautern, 67663 Kaiserslautern, Germany

[2] Image Processing Department, Fraunhofer ITWM, 67663 Kaiserslautern, Germany

[3] Department of Computer Science, University of California, Davis, CA 95616, USA

Published online: 15 October 2020

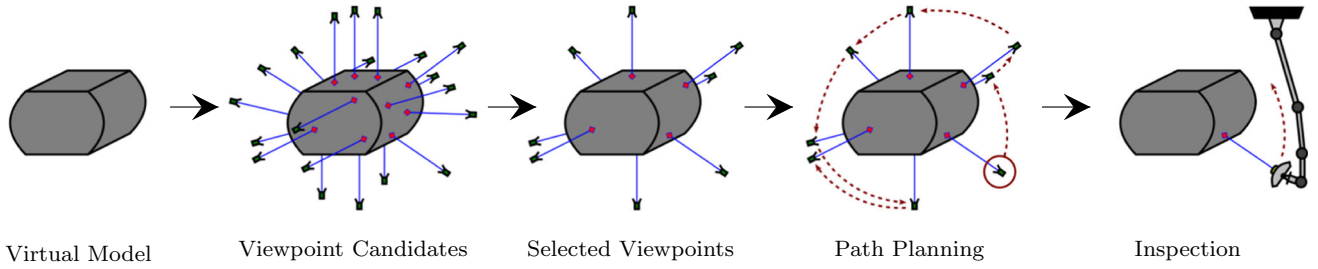


Fig. 1 Overall pipeline. A set of viewpoint candidates gets produced by placing virtual cameras and directing them at certain pivot points on the objects surface. From these candidates, a lower number of final viewpoints get selected so that the object is still sufficiently covered. Then, a camera path is planned considering the degrees of freedom of the used inspection system. Finally, the physical inspection system carries out the inspection by moving the camera along the computed trajectory and taking pictures at the selected viewpoints

the product geometry and its reflection properties, *construction* of a prototype configuration, *verification* using several machine vision algorithms, and *configuration adjustment*. It is a process which works well, given a low geometric complexity of the surface. An increase in complexity makes the system design more tedious and harder to evaluate in terms of inspection completeness and accuracy, thus raising the need of developing an automated design (planning) tool. With general shift of the industry toward more flexible production lines producing small size batches of geometrically very different objects, this challenge becomes more relevant. For example, additive manufacturing (3D printing) is one important application, where no adaptive inspection solution yet exists.

During the assessment phase, an engineer will observe the product from various angles, learning about object's characteristics (features) such as visible or potentially occluded surfaces (geometry) and light response behavior. Based on the observed features and own experience, the engineer will infer the initial acquisition system configuration, relative to the inspected object, and introduce further refinement. Even for an expert, this process is often based on trial and error and requires many iterations and tests. As a consequence, setting up an inspection system manually can take days or even weeks until it meets the required quality criteria. The described process can be translated into more formal steps: (1) read object features, (2) consider potential configurations, (3) choose the most promising configuration (based on object features and expert knowledge). The first two steps belong to object space exploration and the third to optimization. Once a set of optimal viewpoints (camera positions) is determined, the physical system prototype has to be configured. Cameras must be placed either statically or dynamically using a manipulator. In case of a manipulator, an additional path planning step is required for viewpoint traversal. Figure 1 illustrates this pipeline.

In this paper, we focus on the object space exploration, i.e., the generation of viewpoint candidates. Our approach is based on an analytic description of a 3D model using B-spline surfaces. We define a number of *feature functionals* that measure inspection-relevant features on an object's surface. The viewpoint candidates are derived via non-uniform sampling driven by the feature functionals. To validate the quality of our results, we perform a standard algorithm for the next step in the overall inspection pipeline, i.e., we select an optimal subset of viewpoints using a *next-best-view* greedy algorithm. It is a straight-forward algorithm which performs well and provides good results for this particular purpose, as stated by Gronle et al. [11]. The optimal viewpoints generated by this approach are qualitatively very similar to those produced by equivalent approaches. However, since the underlying set coverage problem is NP-hard, the algorithm strongly benefits from the fact that our method produces a small number of viewpoint candidates.

2 Related work

The need for automated inspection planning tools was recognized decades ago, motivating many research efforts in the 1990s and early 2000s. Since then, automated inspection has received less attention.

Early work of Cowan and Kovesi [5] presents an analytical approach, suggesting that for each surface polygon, a 3D solution space should be formed by solving inspection requirements posed as analytic constraints. The optimal solution is then reached through intersection of all obtained solution spaces. Their work focuses on a low-complexity surface patch, since such an approach is computationally expensive.

To reduce the complexity of inspection planning, many subsequent contributions aimed at *good, practically acceptable* solutions instead of *optimal solutions*. A common approach samples the space around the object and uses the model only to evaluate or validate resulting viewpoint candidates.

For example, Sakane et al. [20,21] use uniformly and adaptively tessellated spheres for solution space discretization and further evaluate the sphere facets by two criteria: reliability

for recovering the surface normal vectors and detectability of the surface by camera and illuminators. The size of the viewpoint candidate sets is briefly commented on, stating that the desired number of viewpoint candidates should be manually chosen to balance the trade-off between planning complexity and accurate inspection representation.

Tarbox and Gottschlich [28] use a densely sampled viewing sphere, explicitly stating they have no apriori preference on viewpoint placement. The camera viewing distance d is fixed to an arbitrary number, and the viewpoints are uniformly distributed over the sphere's surface. It is assumed that the visible parts of the object are always within the acceptable depth of field. The main incentive for using dense sampling is to increase the likelihood of the discovery and coverage of regions which are difficult to sense by introducing redundancy. For meshes containing 1000–2500 vertices, they produce a set of 15,000 viewpoint candidates which is then used for the evaluation of the proposed optimization algorithms. They stress discrete surface representation as a major drawback of the sensor planning algorithms because the allowed shapes of the surfaces are restricted.

Tarabanis et al. contribute to the inspection planning field through their work on the MVP (machine vision planner) system [26,27], where the use of viewpoint candidate sets is recognized as the *generate-and-test* approach to inspection planning. The survey considers objects containing only polyhedral features, and the authors state a clear need for the development of planning algorithms capable of handling complex surfaces. They are inclined toward the *synthesis approach*, stating that the viewpoint candidate approach might have the following drawbacks: computational costs of the combinatorial search over finely tessellated (spherical) parameter spaces, use of a tessellated sphere for viewpoint visibility evaluation, and overall scalability of the approach.

Jing [14] employs a mesh dilation by a maximum depth field within which a camera can operate. Such an approach resembles the tessellated sphere, but with higher correlation with the model geometry. The viewpoint candidates are obtained by sampling the dilated surface until a predefined number of viewpoint candidates are reached. Their orientation is calculated based on the distance d to the original object primitives, where each primitive has an attraction force of $1/d^2$.

More recent contributions use a given 3D model actively. Instead of focusing on the space around the object, they directly sample the surface and determine camera positions with a certain distance to the surface points. Such approaches are more appropriate for the computation of feature-sensitive results.

In his work on underwater inspection path planning, Englot [8] creates a viewpoint candidate set using random sampling of the mesh primitives, where each primitive is used as a pivot point for a viewpoint. The viewpoints are generated until each facet of the model has reached an arbitrary level of redundancy (number of different viewpoint candidates from which a facet can be observed), resulting in ca. 2500 and 1300 viewpoint candidates from approximately 131,000 and 107,000 vertex meshes, respectively.

Sheng et al. [23,24] focus on inspecting objects consisting of surfaces with prominent differences in orientation or having low degree of waviness. They take the *flat patch growing* approach by first splitting the surface into sub-patches, calculating the bounding box of each sub-patch, and finally placing the viewpoint candidate at a fixed distance along the normal of the largest bounding box side.

Prieto et al. [19] discretize a 3D model described by NURBS into voxels, split the object into subsurfaces representing simple patches, and, for each patch, find a viewpoint which can be used on the surface in a sweeping manner (also known as view swaths). While such an approach works well on simple objects, it struggles when applied on free form or highly complex surfaces.

Scott [22] suggests an alternative model-based approach using geometric information of the object. A viewpoint candidate is created for each vertex of the mesh, which is stated to be the optimal point for surface description. Due to the high resolution and geometric complexity of some discrete models, the first step is to perform a curvature-sensitive resampling. After an additional decimation step, the final viewpoint candidate set is created. A good decimation level is heuristically determined to be 32 times lower than the target model. The method achieves good results for a set covering approach (e.g., a model of ca. 36,000 vertices is resampled to 702 vertices and decimated to 72 viewpoint candidates). The introduced geometry-based exploration principles provide the foundation for the feature-driven approach presented in this paper.

Gronle and Osten [11] agree with [22] on generating one viewpoint candidate per mesh vertex. Instead of sampling down the triangulation, they reduce the cardinality of their viewpoint candidate sets using an octree. Each leaf contains a group of neighboring candidates from which random viewpoints get selected while others with similar orientation get deleted.

Beside discrete solutions, recent works by Mavrinac et al. [16] and Mohammadikaji [17] show a rise in continuous optimization, which does not require the generation of a viewpoint candidate set. It is a possible alternative to inspection planning and enables the discovery of a truly optimal solution at the cost of higher computational requirements. Due to the different nature of the general approach, an in-depth comparison is beyond the scope of this paper.

The viewpoint candidate set generation itself has not received much attention as a solemn research focus so far. While it is present in the available literature, various approaches and their impact to the selected final viewpoints

are not discussed. Researchers mostly approach it in a way to obtain any kind of reasonable search space before proceeding onto the optimization which is the focus of their work. The obtained viewpoint candidate set is assumed to be dense enough to contain the optimal viewpoints and is thus deemed adequate. While such approaches offer a rather clean shortcut to tackling the optimization problem, they also inherently impose an optimization handicap due to the fact that viewpoint candidates are mostly not generated based on geometric features of the model. The object space exploration approaches used so far are reasonable and good in terms of implementation simplicity. However, they also do not revise different possibilities or modifications which could produce both more meaningful and application-specific results, as well as reduce cardinality of the generated viewpoint candidate set. Existing solutions have not considered uneven object space sampling by concentrating the viewpoint candidates in areas which might appear challenging during the inspection based on various criteria. Such a modification is crucial due to the fact that the optimization problem at hand is a set covering problem and thus NP-hard. As such, it requires a combinatorial approach to solve it. This work aims to decrease the combinatorial complexity challenge by providing an adaptive method for feature-driven placement of viewpoint candidates. The results are meaningful and application-specific viewpoint sets of suitable size for the subsequent optimization process.

3 Method

The novel strategy, presented in this section, finds viewpoint candidates by subdividing a continuous model of parametric surfaces into smaller segments, leading to an adaptive and feature-based sampling of the object.

We define a *viewpoint candidate* by camera-specific data, i.e., the camera's position in world coordinates and the view vector defining the camera's line of sight.

The approach uses a parametric surface representation; in particular, it is designed for (but not limited to) objects described by B-spline surfaces. While those surfaces are often an intermediate product of the virtual design process, they are rarely available to inspection system engineers. Hence, it is assumed that only a triangulated surface mesh of the object of interest is given. This can either be a digitally designed model or a result of a 3D reconstruction of a physical object. Due to the use of lasers, 3D scanning tools produce output in the form of point clouds; a conversion to a triangle mesh can easily be achieved by a variety of freely available tools and algorithms, e.g., [4]. When a model is given only in such a discrete format, we compute a set of continuously connected B-spline surfaces approximating the original data in a first step. Of course, the input data—containing a low or high degree of noise—have direct impact on the shape quality of the computed B-spline surface model. If the input data are obtained via 3D scanning, for example, then it will be essential to ensure highly accurate scanning with a low degree of noise to obtain high-quality B-spline models.

Using B-splines makes the subsequent steps fully independent of the resolution of the input mesh and supports the computation of analytically describable measurements on the surfaces without undesirable discretization effects. Given a model, the user needs to specify a so called *feature functional* E that measures the relevance of a surface region with respect to certain application-specific features. It should be designed in such a way that it has a high value for regions that should be densely covered by viewpoint candidates and a low value in less relevant regions. Let S be a parametric surface, parameterized over a rectangular region $\Omega_0 = [u_{\min}, u_{\max}] \times [v_{\min}, v_{\max}] \subset \mathbb{R}^2$. A point on the surface is denoted by $S(u, v) \in \mathbb{R}^3$, and the notation $S(\Omega) = \{S(u, v) | (u, v) \in \Omega\}$ is used to describe the segment of the surface corresponding to the region $\Omega \subseteq \Omega_0$. The value of the feature functional for the entire surface is denoted by $E(S)$ and the value of the segment corresponding to a parameter region Ω by $E(S, \Omega)$.

In order for the algorithm to converge, the values of the $E(S, \Omega)$ need to converge to zero as Ω converges to a point. To obtain a stable subdivision behavior with respect to the chosen termination threshold, the value of the feature functional computed for part of a region needs to be smaller than the value computed for the region itself, i.e., for two parameter regions $\Omega, \Omega' \subseteq \Omega_0$, it is required that $E(S, \Omega') < E(S, \Omega)$ if $\Omega' \subset \Omega$.

For functionals that are defined as integrals over some function f of the form

$$E(S, \Omega) = \iint_{\Omega} f(S, u, v) \mathrm{d}u \mathrm{d}v,$$

it follows that

$$E(S, \Omega) = E(S, \Omega') + E(S, \Omega'')$$

when $\Omega' \cup \Omega'' = \Omega$ and $\Omega' \cap \Omega'' = \emptyset$. In this case, E is said to be *additive*, which means that partitioning a surface segment leads to a partitioning of the evaluated values. This leads to a monotonic decrease when the surface is subdivided. When the function f has a regular distribution of values, the value of E will decrease approximately exponentially with respect to the depth of the subdivision.

The chosen measurement is used to steer the placement and distribution of viewpoint candidates by recursively subdividing regions with high values until a pre-defined threshold is met. Once the surface network is subdivided,

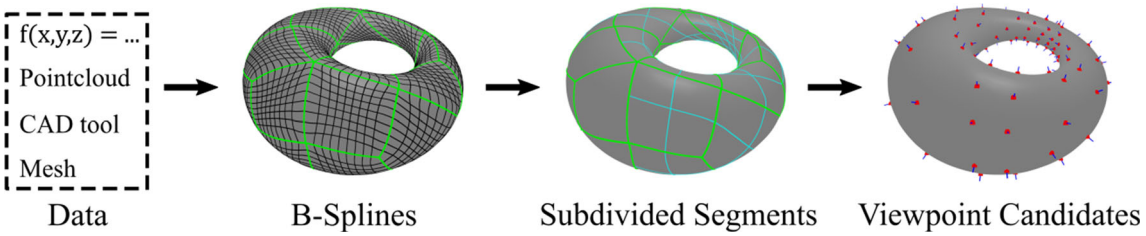


Fig. 2 Generating viewpoint candidates. Discrete data are approximated by a B-spline model. Surfaces are subdivided according to a user-defined geometric feature functional (e.g., curvature). Viewpoint candidate pivot points are placed at the centers of each subdivided segment. The cameras are positioned on the line defined by the normal vectors of the surface

a viewpoint candidate is computed for each segment of the subdivided model.

Assuming that the initial input is a triangle mesh model, the individual steps can be summarized as follows:

1. Generate a representation of the object with a finite number of B-spline surfaces.
2. Subdivide the B-spline surfaces with respect to a user-defined functional $E(S, \Omega)$ and an associated threshold value t.
3. Place viewpoint candidates in $\mathbb{R}^3$ based on the subdivision of the model.

Figure 2 illustrates that process. Sections 3.1, 3.2, and 3.3 explain the individual steps of this strategy in detail. Section 4 discusses a variety of possible and practical feature functionals.

3.1 Obtaining a B-spline model

A B-spline surface is defined by several variables. For a given order $k \in \mathbb{N}$, a (quadratic) grid of $n_c \times n_c$ control points $b_{i,j} \in \mathbb{R}^3$, and a knot vector $\tau \in [0, 1]^{n_c+k+1}$, it is denoted by

$$S : [0, 1]^2 \to \mathbb{R}^3$$
$$S(u, v) = \sum_{i=1}^{n_c} \sum_{j=1}^{n_c} N_{i,k,\tau}(u) N_{j,k,\tau}(v) b_{i,j} \quad (1)$$

where $N_{i,k,\tau}(\cdot)$ is the i-th B-spline basis function of order k and associated knot vector τ.

The B-splines used in the context of this paper are cubic (order $k = 3$) and have uniform knot vectors with multiple end knots

$$\tau = (\underbrace{0, \ldots, 0}_{k+1 \text{ times}}, \tau_1, \ldots, \tau_{n_c-k-1}, \underbrace{1, \ldots, 1}_{k+1 \text{ times}})$$

with $\tau_i = \frac{i}{n_c-k}$ for $i = 1, \ldots, n_c - k - 1$. In general, the presented method is not restricted to this choice.

A single B-spline surface is usually not sufficient to represent a complex object as it can support only disk-like surface topology. Therefore, a number of B-spline surfaces are needed, each modeling a part of the object, and they are stitched together to define an overall watertight (C^0-continuous) model.

Constructing such a network of B-spline surfaces to represent discrete data has been researched for many years and is well covered in the literature [9]. Figure 3 illustrates the general idea of obtaining such a network. The remainder of this section summarizes the individual steps and basic concepts.

In order to compute such a network of B-splines to approximate a triangulated surface, the surface first needs to be subdivided into a collection of four-sided regions (quadrilaterals). For small or geometrically simple objects, this can be done manually, e.g., by defining a set of cutting planes or simply drawing boundary curves on the surface using CAD tools.

When objects are too complex to be processed manually, an automatic method is needed. Many solutions are available. Here, an approach described by Eck and Hoppe [7] is used. This method starts by computing a surface Voronoi tessellation, ensuring that all cells have disk-like topology. Then, the dual Delaunay-like complex is constructed, consisting of triangular cells. Finally, these cells are merged pairwise into quadrilateral cells. While this is an older approach, it still provides good results and can be implemented easily. When it is necessary to deal with more complex topological situations, various advanced algorithms and their implementations are available as well [13].

To set the discrete data points inside each quadrilateral (quad) cell in correlation with the respective parametric surface, the cells are parameterized over the unit square, i.e., each vertex of the triangulated mesh p_i in a cell is assigned a parameter value $(u_i, v_i) \in [0, 1]^2$. Various methods for this step are available. For example, CGAL [29] provides a package for mesh parameterization.

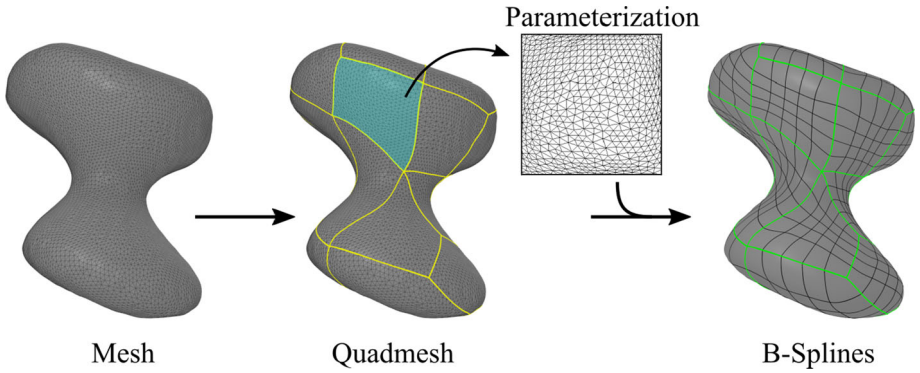


Fig. 3 Construction of a B-spline model approximating a triangulated surface. First, the mesh is partitioned into a collection of quadrilateral cells. Each cell is parameterized over a square region providing the correlation between discrete data points and their location on the B-spline surface. The B-spline model is obtained by minimizing the distances of those points using a least-squares approach

A B-spline surface S that approximates the data points within a quad cell can be computed by minimizing the least-squares error E_{LS}, given by

$$E_{\mathrm{LS}} = \sum_i \|S(u_i, v_i) - p_i\|^2. \tag{2}$$

The only variables in this term are the B-spline control points $b_{i,j}$. Minimizing (2) results in a surface that approximates the discrete data points as closely as possible, ignoring the spaces in between. As a consequence, it often includes unwanted wiggles that are not part of the original surfaces. This effect can be mitigated by adding a fairing term that penalizes this effect. A commonly used approach is the *thin plate energy* functional [7,12] given by

$$\begin{aligned} E_{\mathrm{TP}} = \int_0^1 \int_0^1 & \|\partial_{uu} S(u, v)\|^2 \\ & + 2\,\|\partial_{uv} S(u, v)\|^2 \\ & + \|\partial_{vv} S(u, v)\|^2 \,\mathrm{d}u\mathrm{d}v. \end{aligned} \tag{3}$$

A smooth surface can be computed by minimizing

$$(1 - \lambda) E_{\mathrm{LS}} + \lambda E_{\mathrm{TP}} \to \min \tag{4}$$

with $\lambda \in [0, 1)$. A high value of λ leads to an approximation with a mostly near-planar, averaging surface, while a low value causes the surface to be closer to the data points.

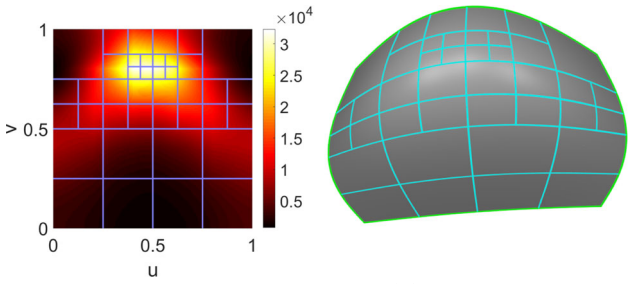
A C^0-continuous network of surfaces can be obtained by adding a set of linear constraints that force control points of neighboring surfaces along their shared boundary curve to be equal.

While the generation of B-spline models is not our focus, it is a relevant step when only discrete input data are available. To ensure the desired behavior of subsequent data processing steps, it is important that the B-spline surfaces capture relevant features of the real-world object well, e.g., overall shape, curvature, and topology. Consequentially, this requirement must be satisfied, to a lower degree, by the discrete input data as well. The combination of least-squares approximation and smoothing (4) prevents single deviating data points from having a notable impact to results. However, the quality of the B-spline model strongly decreases when exhibiting significant noise or incorrect topology—problems that can arise when using low-resolution 3D scanning technology.

3.2 Subdivision of the model

This section describes the feature-based subdivision approach for a given B-spline model consisting of n_s surfaces $S_i : \Omega_0 \to \mathbb{R}^3, i = 1, \ldots, n_s$ defined over a rectangular parameter region $\Omega_0 \subset \mathbb{R}^2$. Let $E(S, \Omega)$ be the given functional that measures how prominent a feature of interest occurs within the segment of S restricted to $\Omega \subset \Omega_0$. The goal is to subdivide each surface into segments, such that E, evaluated for any of these segments, is below a given threshold t. Since the presented recursive approach considers each surface individually, it can be applied in parallel to all B-spline surfaces of the model.

Subdivision of the B-splines takes place in parameter space. The subdivision method is inspired by the work of Nouanesengsy et al. concerning analysis-driven refinement [18]. Initially, for each B-spline surface S, the given functional E is evaluated for the entire surface. If its value is larger than t, the parameter domain is subdivided into four equal-size rectangles. The procedure continues recursively for each subregion. Once the value of E evaluated for a segment falls below t, the corresponding parameter region does not get subdivided any further. Algorithm 1 summarizes this procedure. Figure 4 demonstrates the subdivision of a single surface based on thin plate energy.



(a) Subdivision in parameter space **(b)** B-Spline surface in $\mathbb{R}^3$

Fig. 4 Subdivision of a single B-spline surface using thin plate energy as feature functional. Plotting the integrand value of Eq. (3) in the parameter space $\Omega_0 = [0, 1] \times [0, 1]$ illustrates how the approach leads to a deeper subdivision in areas with high values (**a**). **b** Shows the surface $S(\Omega_0)$ and its subdivision into segments in $\mathbb{R}^3$

Algorithm 1 Recursive Subdivision

function SUBDIVIDESURFACE(S, Ω, E, t)
  if $E(S, \Omega) > t$ **then**
    $(\Omega_1, \Omega_2, \Omega_3, \Omega_4) \leftarrow$ SPLIT(Ω)
    return $\bigcup_{i=1}^{4}$ SUBDIVIDESURFACE(S, Ω_i, E, t)
  else
    return $\{S(\Omega)\}$
  end if
end function

function SPLIT($\Omega = [u_0, u_1] \times [v_0, v_1]$)
  $u_c \leftarrow \frac{u_0+u_1}{2}$
  $v_c \leftarrow \frac{v_0+v_1}{2}$
  $\Omega_1 \leftarrow [u_0, u_c] \times [v_0, v_c]$
  $\Omega_2 \leftarrow [u_c, u_1] \times [v_0, v_c]$
  $\Omega_3 \leftarrow [u_0, u_c] \times [v_c, v_1]$
  $\Omega_4 \leftarrow [u_c, u_1] \times [v_c, v_1]$
  return $(\Omega_1, \Omega_2, \Omega_3, \Omega_4)$
end function

3.2.1 Choosing the subdivision threshold

Finding a good subdivision threshold value depends on the understanding of the inspection requirements and intuition behind the used subdivision criteria. For example, a measurement that computes a maximum incidence angle for a segment is easily understood, while an integral of principal curvatures requires the user to have specialized training in geometry in order to interpret the resulting values.

Defining the subdivision threshold manually requires technical knowledge about the expected values for differently shaped surfaces. This can be mitigated by evaluating the feature functionals for all individual surfaces and calculating the average value

$$t_{\text{avg}} = \frac{\sum_{i=1}^{n_s} E(S)}{n_s}. \tag{5}$$

The user defines a percentage of the average to control the amount of viewpoint candidates. For example, choosing $t = 0.5t_{\text{avg}}$ leads to a subdivision of the surface network until the values of each patch are below half of the overall average feature value. Alternatively, if the feature functional has the additivity property, it is possible to express the threshold as a percentage of the total value over the entire object.

A suitable subdivision threshold can also be chosen interactively in real time. In this scenario, it is necessary to pre-compute the subdivision for a small threshold t_1, e.g., a lower bound for the range of considered thresholds. Depending on the complexity of the feature functional, computing this fine subdivision can be time-consuming. However, when all intermediate steps and evaluated measurements are stored in a tree structure, the tree can be cut back in real time for any threshold $t_2 > t_1$. This makes it possible for the system engineer to adjust a slider in a graphical user interface and interact with a visual representation of the subdivided model to determine the best threshold for a specific application scenario.

3.2.2 Modifications

If the used feature functional is additive, it allows for a potential speedup of the process by starting with a fine subdivision of all surfaces. The subdivision tree is then computed backwards, i.e., by merging individual surface segments as long as the value of the merged patch stays below the subdivision threshold. The computational speedup is due to the fact that E only has to be evaluated for the deepest subdivision level. The values of the combined segments can be obtained by adding the values of the individual segments.

Furthermore, the algorithm can be extended to handle multiple functionals describing different features of interest, e.g., both curvature and surface area. For this purpose, the feature functional and threshold parameter must be replaced by a list of functional threshold pairs. All functionals are evaluated, and when one exceeds its corresponding threshold, the evaluated segment is subdivided.

3.2.3 Iterative subdivision

The goal of the recursive subdivision is to generate a sufficient sampling of the model with respect to the features of interest. Alternatively, a global approach guided by a desired number of viewpoint candidates can be chosen. This approach becomes necessary when external constraints limit the amount of allowed viewpoint candidates.

Let t_n be a given number of maximum viewpoints for the entire model. Instead of individually subdividing each surface until the measurement reaches the subdivision threshold, the feature functional is evaluated for all surface segments and the segment where it is maximal is subdivided. This procedure is repeated until the desired amount of segments is reached. Algorithm 2 summarizes this approach.

Algorithm 2 Iterative Subdivision

function SUBDIVIDEMODEL($\{S_1, S_2, ..., S_{n_s}\}, E, t_n$)
  $\mathcal{S} \leftarrow \{S_1, S_2, ..., S_{n_s}\}$
  while $|\mathcal{S}| < t_n$ **do**
    $(S', \Omega') \leftarrow \underset{S,\Omega:S(\Omega)\in\mathcal{S}}{\text{argmax}} \; E(S)$
    $(\Omega_1, \Omega_2, \Omega_3, \Omega_4) \leftarrow \text{SPLIT}(\Omega')$
    $\mathcal{S} \leftarrow \mathcal{S} \setminus S'(\Omega') \cup \{S'(\Omega_1), S'(\Omega_2), S'(\Omega_3), S'(\Omega_4)\}$
  end while
end function

3.3 Computing the viewpoint candidates

After the subdivision phase, one viewpoint candidate is placed per surface segment of the subdivided model. First, a pivot point is chosen, i.e., the point that determines the camera viewing direction. Then, the actual camera position is calculated via a user-defined distance d. This distance is manually determined by a system engineer according to the inspection setup requirements. That way, the viewpoint candidate generation approach assures compliance with the camera's depth-of-field constraints at generation time.

The approximated center of the segment is used as a pivot point. It can be obtained by evaluating the surface at the center point of its respective parameter domain. The camera position is computed by moving away from the surface in the direction of the surface normal, given by

$$n(u, v) = \frac{\partial_u S(u, v) \times \partial_v S(u, v)}{\|\partial_u S(u, v) \times \partial_v S(u, v)\|}.$$

The position of the camera is defined as

$$C^d_{S,\Omega} = S(u_c, v_c) + n(u_c, v_c) \cdot d, \tag{6}$$

where (u_c, v_c) is the center of the parameter domain Ω. For a rectangular region $\Omega = [u_0, u_1] \times [v_0, v_1]$, it is given by $u_c = \frac{u_0+u_1}{2}$ and $v_c = \frac{v_0+v_1}{2}$. The view direction of the camera is the reversed normal vector. Figure 5 illustrates this concept.

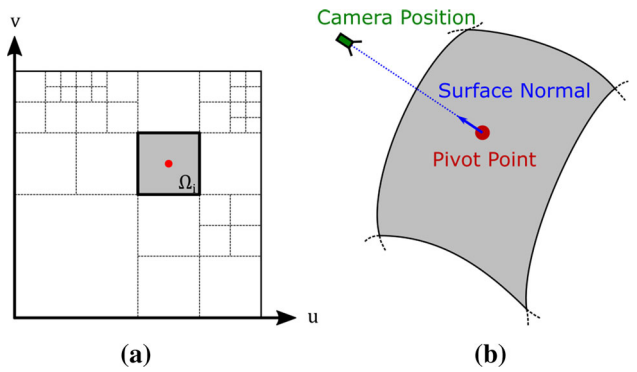


Fig. 5 Computing the viewpoint candidate for a surface segment. **a** Illustrates a subdivision of a surface in parameter space. The image $S(\Omega_i)$ of the highlighted region is shown in (**b**). Evaluating the surface at the center of Ω_i yields the pivot point for this segment. The camera position is obtained by moving away in the direction of the surface normal at that point

4 Feature functionals

The feature functional E is essential for distributing the viewpoint candidates. In order to produce high-quality results, tailored to specific applications, it needs to be defined in a way that identifies areas of interest by assigning them high values while at the same time giving low values to areas that do not contain the desired features. It is up to the system engineer to either choose or define a functional applicable to their use case. A good functional should be intuitive and easy to compute.

As discussed in Sect. 3, the values of $E(S, \Omega)$ need to converge toward zero as Ω converges to a single point to ensure termination of the subdivision. The following subsections introduce a number of fundamental feature functionals that satisfy this property. They are suitable candidates for a variety of shapes, as they highlight common geometric features of interest. The modularity of the algorithm allows it to be easily extended with custom functionals as long as they also satisfy the convergence property.

Some of the presented feature functionals are defined as integrals. For implementation purposes, those can be evaluated numerically. Trapezoidal rules have proved to be an adequate method, as for a sufficiently dense sampling of the surface, the error becomes neglectably small [1].

4.1 Curvature

In the context of surface inspection, the placement of viewpoint candidates should be focused around highly curved parts of the object. The high surface variation in those regions usually requires an inspection from several directions, while flat regions can be covered with a single or only a few viewpoints.

Commonly used terms to describe curvature are the *principal curvatures* κ_1 and κ_2. At any given point on the surface, these are the normal curvature values in the directions in which the surface has its lowest and highest bending, respectively. Combined, they define the mean curvature $H = \frac{\kappa_1+\kappa_2}{2}$ and the Gaussian curvature $K = \kappa_1\kappa_2$. More detailed information on differential geometry and curvature can be found in literature [6].

Individually, these terms are not suited to separate flat and curved regions, as both can become zero in non-flat regions, i.e., if one of the principal curvatures is zero while the other has a value other than zero (Gaussian curvature)

or if $\kappa_1 = -\kappa_2 \neq 0$ (mean curvature). This can be resolved by directly considering the principal curvatures. The surface integral given by

$$E_\kappa(S) = \iint_S \kappa_1^2 + \kappa_2^2 \, \mathrm{d}S \tag{7}$$

is a meaningful measurement to describe the total bending of a surface S [12]. It is equal to zero if and only if S is a flat surface. To compute the integral with respect to a parameter region Ω, it can be rewritten as

$$E_\kappa(S, \Omega) = \iint_\Omega (\kappa_1^2(u, v) + \kappa_2^2(u, v)) \, \|\partial_u S(u, v) \times \partial_v S(u, v)\| \, \mathrm{d}u\mathrm{d}v. \tag{8}$$

B-splines are parametric surfaces with well-defined first and second derivatives everywhere. Hence, it is possible to analytically evaluate κ_1 and κ_2 at any point and numerically compute the entire integral.

For implementation purposes, it is useful to express the integrand of Eq. (8) in terms of Gaussian and mean curvature:

$$\kappa_1^2 + \kappa_2^2 = 4H^2 - 2K.$$

Evaluating this term requires less operations, as H and K can be obtained directly from the coefficients of the first and second fundamental forms which are straightforward to compute at any point of a B-spline surface [6].

4.2 Thin plate energy

Numerically computing the integral of the principle curvatures yields precise and intuitive results at the price of high computational cost. However, it can be sped up immensely by approximating Eq. (7) using the thin plate energy functional, as introduced in Eq. (3).

For B-splines, this term can be rewritten in a more compact form by expressing the grid of $n_c \times n_c$ control points $b_{i,j}$ as one-dimensional list. Let $b_x, b_y, b_z \in \mathbb{R}^{n_c^2}$ be column vectors containing the x, y, z components of the control points. Equivalently, let $N(u, v) \in \mathbb{R}^{1 \times n_c^2}$ be a row vector, containing the corresponding products of basis functions $N_{i,k,\tau}(u) N_{j,k,\tau}(v)$. Using this notation, a B-spline surface (1) can be rewritten as

$$S(u, v) = (N(u, v)b_x, N(u, v)b_y, N(u, v)b_z)^{\mathrm{T}}.$$

Then, the thin plate energy can be expressed as

$$\begin{aligned} E_{\mathrm{TP}}(S, \Omega) = \iint_\Omega & \|\partial_{uu} S(u, v)\|^2 \\ & + 2 \|\partial_{uv} S(u, v)\|^2 \\ & + \|\partial_{vv} S(u, v)\|^2 \, \mathrm{d}u\mathrm{d}v \\ = & \, b_x^T M b_x + b_y^T M b_y + b_z^T M b_z \end{aligned} \tag{9}$$

with the matrix $M \in \mathbb{R}^{n_c^2 \times n_c^2}$ given by

$$\begin{aligned} M = \iint_\Omega & (\partial_{uu} N(u, v))^T (\partial_{uu} N(u, v)) \\ & + 2(\partial_{uv} N(u, v))^T (\partial_{uv} N(u, v)) \\ & + (\partial_{vv} N(u, v))^T (\partial_{vv} N(u, v)) \mathrm{d}u\mathrm{d}v. \end{aligned}$$

Since the basis functions are piecewise polynomials, the integral can either be computed analytically or by integrating each polynomial segment using a numeric integration rule, e.g., Gaussian quadrature with an order sufficiently high to be exact for polynomials of the given degree. After calculating M, evaluating $E_{\mathrm{TP}}(S, \Omega)$ requires the calculation of computationally inexpensive matrix–vector products.

The coefficients of the matrix M only depend on the used knot vector τ, the order k of the B-spline, and the evaluated parameter region Ω. Assuming that all B-spline surfaces in the model have the same order and knot vector, M only needs to be computed once per subregion $\Omega \subseteq \Omega_0$ that is considered during a subdivision step. The matrices for individual subregions can be precomputed and saved up to a predefined maximum subdivision depth allowing for them to be reused on different models with same order and knot vector.

4.3 Maximum deviation from average normal

An alternative approach to obtain a subdivision into mostly flat surface segments is to aim at a low variation in normal vectors within each patch. Finding the maximum angle between normal vectors at any two arbitrary surface points is a nontrivial task. A more practical solution is to calculate the average normal of a surface segment and compute the maximum angle between any normal vector on the segment and this average normal. This is a modification to the approach used by Sheng et al. [23] who apply this concept in the discrete case of triangulated surfaces.

In the continuous setting of B-splines, the average normal vector of a surface segment is given by

$$\tilde{n}_{\mathrm{avg}}(S, \Omega) = \frac{\iint_\Omega n(u, v) \, \|\partial_u S(u, v) \times \partial_v S(u, v)\| \, \mathrm{d}u\mathrm{d}v}{\iint_\Omega \|\partial_u S(u, v) \times \partial_v S(u, v)\| \, \mathrm{d}u\mathrm{d}v}$$

with normalization

$$n_{\mathrm{avg}}(S, \Omega) = \frac{\tilde{n}_{\mathrm{avg}}(S, \Omega)}{\|\tilde{n}_{\mathrm{avg}}(S, \Omega)\|}.$$

The functional to measure maximum normal deviation is then defined as

$$E_{\mathrm{ND}}(S, \Omega) = \max_{(u,v)\in\Omega} \cos^{-1}\left(n_{\mathrm{avg}}(S, \Omega) \cdot n(u, v)\right).$$

4.4 Maximum incidence angle

While this work focuses on geometric properties of the target objects, it can also utilize additional knowledge from material science. Some properties like surface roughness or reflection behavior require a restriction of the angle under which the surface of the object is inspected. This is achieved by introducing a functional that measures the maximum deviation between the surface normal at any point of the segment and a ray that originates from a camera placed according to Eq. (6) at distance d:

$$E_{\mathrm{IA}}(S, \Omega) = \max_{(u,v)\in\Omega} \cos^{-1}\left(n(u, v) \cdot \frac{C^d_{S,\Omega} - S(u, v)}{\|C^d_{S,\Omega} - S(u, v)\|}\right).$$

More complex constraints like a range between minimum and maximum angle or deviation from an optimal angle can be modeled in a similar way.

4.5 Area

Surface area itself can also be used as subdivision criterion. It makes sure that at least one viewpoint candidate gets generated per predefined maximum amount of area. This is especially useful in cases where features of interest cannot be expressed by simple mathematical functions or in a prototyping process where specific feature functionals are not yet determined. Moreover, inspection pipeline characteristics such as a narrow camera frustum might impose limiting factors on the amount of the surface that can be processed from a single viewpoint. A subdivision by area will always ensure a certain sampling resolution of the surface.

The surface area is given by

$$E_A(S) = \iint_S 1 \mathrm{d}S. \tag{10}$$

To obtain the feature functional for arbitrary segments, Eq. (10) is rewritten as

$$E_A(S, \Omega) = \iint_\Omega \|\partial_u S(u, v) \times \partial_v S(u, v)\| \,\mathrm{d}u\mathrm{d}v.$$

For complex objects, this functional can be used in combination with other geometric feature functionals like curvature to control sampling rate sparsity in flat regions.

4.6 Region of interest

Engineers and inspectors working with production lines can usually predict where defects will occur based on their experience. Even if the inspection is already automated, this kind of knowledge can be obtained by analyzing the distribution of previously detected defects.

If this information is available, the application scientist can define a region of interest (ROI), either directly by using a brush tool on the surface or freely in 3D world coordinates (e.g., by defining a bounding sphere or bounding box) to highlight regions of the object at which the viewpoint candidate distribution needs to be more intense. With

$$f_{\mathrm{ROI}}(x, y, z) = \begin{cases} 1, & \text{if } (x, y, z) \in \mathrm{ROI} \\ 0, & \text{otherwise} \end{cases},$$

the ROI-functional is defined as the surface area that is inside the region of interest:

$$E_{\mathrm{ROI}}(S, \Omega) = \iint_\Omega f_{\mathrm{ROI}}\left(S(u, v)\right) \|\partial_u S(u, v) \times \partial_v S(u, v)\| \,\mathrm{d}u\mathrm{d}v$$

Besides highlighting regions where errors are expected, this functional can also be used to focus on critical parts of the object in which surface defects have a higher significance.

5 Selecting final viewpoints

After a set of viewpoint candidates is computed, a list of optimal viewpoints needs to be selected with the requirement of keeping the number as low as possible while covering the desired parts of the object. For that purpose, the initial mesh model of the object is used to evaluate the viewpoints. Let F be a set containing the n_F surface primitives (triangles) of the mesh, and let a set of viewpoint candidates $V = \{v_1, ..., v_{n_{\mathrm{vpc}}}\}$ be given. The goal is to find a set of optimal viewpoints $O \subseteq V$ which sufficiently covers the object. To represent the coverage relation between viewpoints and the faces of the mesh, a visibility matrix $A \in \{0, 1\}^{n_F \times n_{\mathrm{vpc}}}$ is constructed, with coefficients $a_{ij} = 1$ if and only if viewpoint v_j covers the i-th triangle $f_i \in F$.

Since all sets are discrete and finite, the task is equivalent to the Set Coverage Problem [25] and can be written as

$$\min \sum_{i=1}^{n_{\mathrm{vpc}}} c_i x_i, \quad c_i = \frac{1}{\sum_{i=1}^{n_F} a_{ij}}$$
$$\text{s.t. } Ax \geq b^{(F)}. \tag{11}$$

The binary variables x_i indicate whether the i-th viewpoint candidate is selected, i.e., $O = \{v_i \in V | x_i = 1\}$. The vector $b^{(F)} \in \{0, 1\}^{n_F}$ controls which surface primitives need to be covered by the inspection system. Using $b^{(F)} = (1, ..., 1)^T$ is equivalent to requiring full coverage.

The next best view approach is frequently used to solve this problem [11,14,22]. Since the goal of this paper is to evaluate viewpoint candidate sets, the same approach is used here. That way, comparable results are obtained and the strengths of the presented approach can be discussed. The visibility matrix is built using ray tracing to create the correlation between the primitives and the viewpoints which can observe them. The starting viewpoint is chosen by finding the triangle covered by the least amount of viewpoints. From the set of viewpoints that observe this triangle, the viewpoint which observes the largest part of the object is chosen. Further viewpoints are chosen in an iterative way by always choosing the viewpoint that observes the most uncovered triangles. The process is repeated until all the primitives are covered or there are no more viewpoints available.

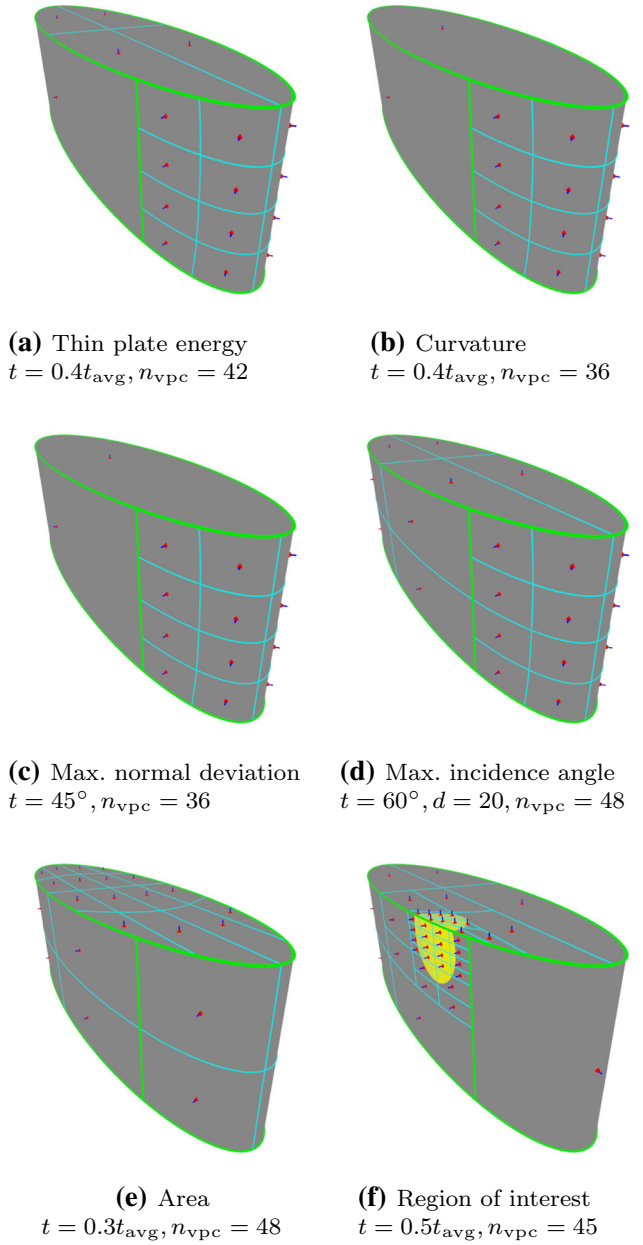
6 Results

The presented method is applied to models of varying complexity to demonstrate the viability of the approach as well as to highlight the differences between the feature functionals introduced in Sect. 4. To enable a meaningful comparison, the subdivision thresholds are chosen to generate approximately the same magnitude of viewpoint candidates.

Each result picture shows the subdivided models including glyphs to mark pivot points and normal directions. When using integral-based functionals, the used subdivision threshold t is given in relation to the average threshold t_{avg} as defined by Eq. (5). For the measurements that are evaluating angles, the threshold is provided as fixed value in degrees. The number of viewpoint candidates generated by the recursive subdivision is denoted by n_{vpc}.

Figure 6 shows the results from applying the recursive subdivision to a cylinder geometry (stretched in one dimension). Top and bottom are each realized by a single B-spline surface while the mantle is modeled by four surfaces. Due to the stretching in x-direction, two of these surfaces are relatively flat, while the other two surfaces are highly bent. The respective sizes of the object in x, y, and z directions are 80, 20, and 20 units in world coordinates.

The spring model shown in Figure 7 is geometrically more difficult. Its minimum bounding box has length and width equal to 15 and height equal to 20 world units. Like the cylinder, it consists of one individual circular surface at the top and bottom, respectively. Instead of a smooth mantle, it consists of more complex surfaces with high variation in curvature along its side. This part is divided into four slices



(a) Thin plate energy $t = 0.4t_{\text{avg}}, n_{\text{vpc}} = 42$

(b) Curvature $t = 0.4t_{\text{avg}}, n_{\text{vpc}} = 36$

(c) Max. normal deviation $t = 45°, n_{\text{vpc}} = 36$

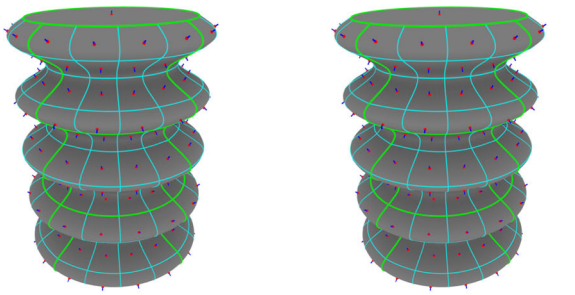
(d) Max. incidence angle $t = 60°, d = 20, n_{\text{vpc}} = 48$

(e) Area $t = 0.3t_{\text{avg}}, n_{\text{vpc}} = 48$

(f) Region of interest $t = 0.5t_{\text{avg}}, n_{\text{vpc}} = 45$

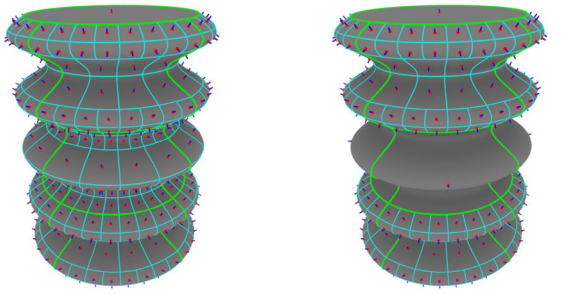
Fig. 6 Application of the recursive subdivision to a cylinder model. The curvature-based functionals **a**–**c** yield a higher sampling where surface bending is high. However, using the thin plate energy also leads to a subdivision of the top and bottom surfaces. Likewise, the incidence angle **d** produces the desired sampling of the curved areas while also subdividing flat regions to provide sufficient coverage under the given angle constraints. The subdivision by area **e** provides a good overall coverage while the region of interest **f** places the viewpoint candidates primarily on the highlighted region

along the circumference, each being split into three individual B-spline surfaces along its height. As long as the goal of the viewpoint placement is only unconstrained coverage, this model can still easily be processed by a human inspector, while the curvature behavior along the mantle introduces difficulties to most automatic methods. However, once con-



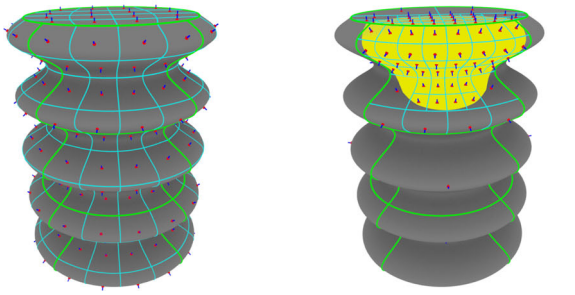
(a) Thin plate energy
$t = 0.2t_{\text{avg}}, n_{\text{vpc}} = 194$

(b) Curvature
$t = 0.2t_{\text{avg}}, n_{\text{vpc}} = 194$



(c) Max. normal deviation
$t = 70°, n_{\text{vpc}} = 482$

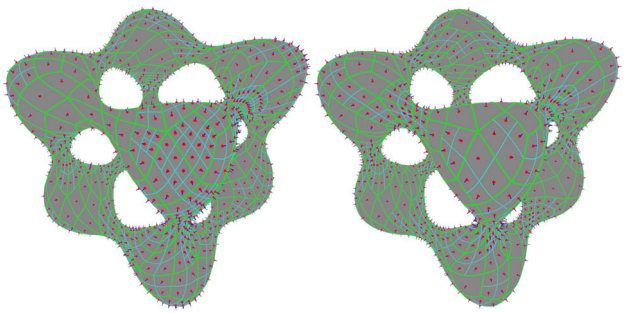
(d) Max. incidence angle
$t = 85°, d = 20, n_{\text{vpc}} = 326$



(e) Area
$t = 0.2t_{\text{avg}}, n_{\text{vpc}} = 224$

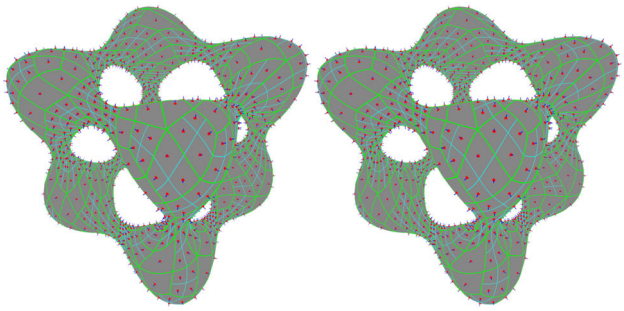
(f) Region of interest
$t = 0.5t_{\text{avg}}, n_{\text{vpc}} = 86$

Fig. 7 Application of the recursive subdivision approach to the spring model. The subdivisions for thin plate energy (**a**) and curvature (**b**) are identical for equivalent thresholds. However, the normal deviation **c** shows less stable results. While the subdivision for the given threshold is finer than for the other curvature-based measurements, choosing a slightly higher threshold would cause no subdivision at all. Using the incidence angle (**d**) is also less stable as the middle part of the object does not get subdivided at all. As in the case of the cylinder model, the area measurement **e** leads to a evenly distributed sampling while the region of interest **f** subdivides only the highlighted part of the model



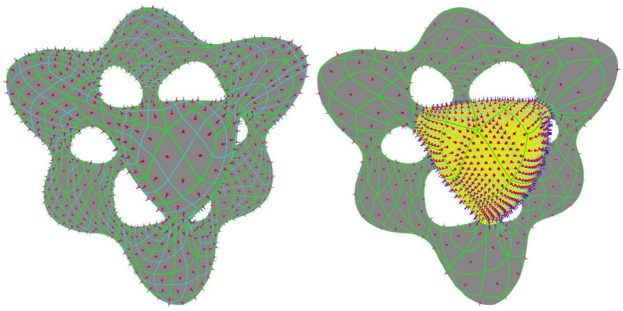
(a) Thin plate energy
$t = 0.75t_{\text{avg}}, n_{\text{vpc}} = 1\,400$

(b) Curvature
$t = 0.75t_{\text{avg}}, n_{\text{vpc}} = 1\,325$



(c) Max. normal deviation
$t = 30°, n_{\text{vpc}} = 1\,571$

(d) Max. incidence angle
$t = 35°, d = 130, n_{\text{vpc}} = 1\,379$



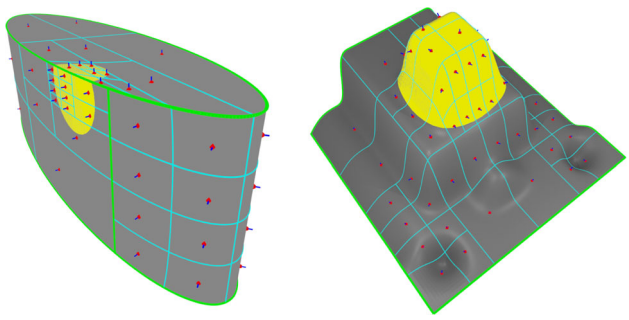
(e) Area
$t = 0.75t_{\text{avg}}, n_{\text{vpc}} = 1\,406$

(f) Region of interest
$t = t_{\text{avg}}, n_{\text{vpc}} = 1\,361$

Fig. 8 When applying the recursive subdivision to more complex models, differences between used functionals become more distinctively visible. Using the thin plate energy **a** leads to a higher refinement on curved regions, but also on some of the flat regions as they still contain planar distortions. Measuring curvature directly **b** provides a more accurate placement of viewpoint candidates around the thin and highly curved parts of the object, while still moderately refining the outer surfaces with medium bending. Similar results are obtained using the normal deviation (**c**) and incidence angle (**d**) functionals. A regular sampling is obtained when considering surface area (**e**). The region-of-interest functional **f** leads to a high intensity of viewpoint candidates in the highlighted part

straints, e.g., coverage under restricted angles, are factored in, the complexity of this task increases drastically for human operators, necessitating an automated solution.

In Figure 8, the recursive subdivision is demonstrated on a geometrically and topologically more challenging object. The data itself had been obtained from a 3D volumetric image, and the B-spline representation with 452 surfaces was computed fully automatically using the surface reconstruction approach discussed in Sect. 3.1. Dimensions of the object are 72 world units in x, y, and z direction, respectively. Set-



(a) $t^{(\mathrm{TP})} = 0.4t_{\mathrm{avg}}^{(\mathrm{TP})}$
$t^{(\mathrm{ROI})} = 0.8t_{\mathrm{avg}}^{(\mathrm{ROI})}$
$n_{\mathrm{vpc}} = 66$

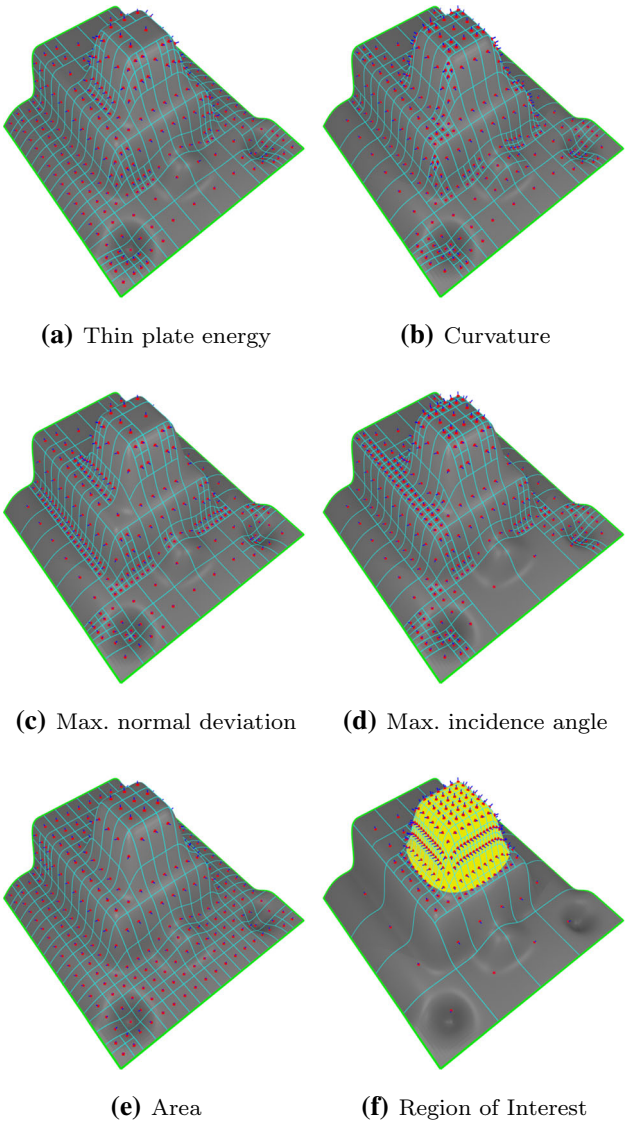
(b) $t^{(\mathrm{TP})} = 0.05t_{\mathrm{avg}}^{(\mathrm{TP})}$
$t^{(\mathrm{ROI})} = 0.1t_{\mathrm{avg}}^{(\mathrm{ROI})}$
$n_{\mathrm{vpc}} = 55$

Fig. 9 Combining feature functionals for the recursive subdivision. Both models are subdivided using both thin plate energy (TP) and region-of-interest functional (ROI), ensuring an adequate overall coverage based on surface bending while also focusing on a highlighted region

ting up an inspection system for this model manually would take considerable effort. In this example, the boundaries of the individual B-spline surfaces are not aligned with any features of the object. Hence, some surfaces can cover both curved and flat parts. The curvature-based criteria are able to handle this scenario especially well, leading to a dense placement of viewpoint candidates around highly bent areas.

In some cases, several types of geometric features are relevant and need to be considered for the placement of viewpoint candidates. As discussed in Sect. 3.2.2, the recursive subdivision approach can evaluate several functionals and compare them with their respective thresholds at each step. Figure 9 demonstrates this modification on selected examples.

Figure 10 shows results of the iterative subdivision (Algorithm 2) applied to a single B-spline surface. This model is 19 world units wide and long while having a height of 11.8 units. It combines a variety of curvature situations within a single surface. In contrast to the recursive approach that focuses on coverage of the object with respect to given objectives, the iterative approach gives the user direct control over the number of viewpoint candidates. Furthermore, it allows for an intuitive comparison of the used feature functionals, as similarities and differences can easily be spotted in the visualizations.



(a) Thin plate energy **(b)** Curvature

(c) Max. normal deviation **(d)** Max. incidence angle

(e) Area **(f)** Region of Interest

Fig. 10 Applying the iterative subdivision constrained by the number of total viewpoints highlights the differences between the presented feature functionals. The number of iterations is limited to 101 leading to 304 viewpoint candidates in total. The curvature-based measurements (**a**–**c**), as well as the incidence angle functional (**d**), focus the subdivision primarily on curved regions. Small differences can be observed on those areas with only moderate bending as the individual functionals prioritize them differently. The area measurement (**e**) provides a regular, feature-independent sampling. Using the region of interest (**f**), subdivisions occur only in segments that contain parts of the highlighted region

6.1 Viewpoint set optimization

To validate the viewpoint candidates generated by our method, we briefly present the results of the subsequent optimization step, i.e., a small number of viewpoints are chosen to solve the set covering problem, see (11). To do so, the viewpoint candidate sets are processed with the next best view approach as explained in Sect. 5. The results are given in Table 1. While the time for the viewpoint candidate generation is only dependent on the properties of the B-splines, the selection step utilizes the original mesh for the raytracing and verification. Hence, the times depend on the number of triangles as well as the number of viewpoint candidates. The machine used to compute these results is equipped with an Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz and uses 16 gigabyte of DDR3 RAM.

Table 1 Results of the viewpoint candidate generation based on the recursive subdivision for different feature functionals and set optimization with goal of maximum coverage

Viewpoint candidate generation				Viewpoint set optimization		
Feature functional	Threshold	SD duration (s)	n_{vpc}	n_{opt}	RT duration (s)	NBV duration (s)
(a) Cylinder (21,800 mesh primitives, 6 surfaces, 16 × 16 control points)						
Thin plate	$0.4t_{avg}$	0.001	**42**	**16**	49.5534	0.522725
Curvature	$0.4t_{avg}$	12.518	**36**	**10**	41.1863	0.42676
Normal deviation	45°	2.848	**36**	**10**	41.3229	0.424306
Incidence angle	60°	4.317	**48**	**22**	56.68	0.578682
Area	$0.3t_{avg}$	2.950	**48**	**24**	61.5702	1.03087
(b) Spring (51,800 mesh primitives, 14 surfaces, 20 × 20 control points).						
Thin plate	$0.2t_{avg}$	0.005	**194**	**13**	267.679	189.91
Curvature	$0.2t_{avg}$	79.921	**194**	**13**	269.428	181.005
Normal deviation	70°	20.835	**482**	**9**	647.222	610.145
Incidence angle	85°	25.189	**326**	**9**	408.495	359.285
Area	$0.2t_{avg}$	17.253	**224**	**13**	312.948	205.012
(c) Tangle cube (82,672 mesh primitives, 452 surfaces, 6 × 6 control points).						
Thin plate	$0.75t_{avg}$	0.008	**1400**	**36**	2614.64	249.015
Curvature	$0.75t_{avg}$	16.696	**1325**	**32**	2556.78	270.601
Normal deviation	30°	2.134	**1571**	**32**	3103.06	289.856
Incidence angle	35°	3.021	**1379**	**33**	1771.99	234.972
Area	$0.75t_{avg}$	1.883	**1406**	**38**	2613.49	268.04

The computational time is caused primarily by three key steps of the process which are measured individually: Application of the recursive subdivision algorithm (SD), computation of the visibility matrix via raytracing (RT), and performing the next best view algorithm (NBV). The number of viewpoint candidates is given by n_{vpc} while n_{opt} denotes the number of viewpoints chosen during the selection step

7 Discussion

As the presented results demonstrate, a good overall functional that provides satisfying results for all types of geometries is difficult to define. However, the fundamental functionals used in this paper have proved to effectively focus the viewpoint candidate placement around the desired features. When a system engineer has a good idea of what characteristics the viewpoint candidates should be focused on, the recursive subdivision will provide the desired result.

The thin plate energy provides good results for most situations. It is also the fastest of the presented functionals and can be evaluated in real time, even for larger objects. As a consequence, it allows the operator to find a good subdivision threshold in an interactive way, e.g., by adjusting a slider and observing the changes in the visualization. Using the thin plate energy, however, can also cause subdivisions in some flat areas. While the subsequent viewpoint optimization for inspection purposes is an NP-hard problem, a few additional viewpoints are usually not an issue. This effect is further analyzed in Sect. 7.1.

Using curvature as subdivision criterion produces dense sampling of highly curved regions without subdividing flat areas at all. Because it requires a high amount of elementary computations, it cannot be evaluated in real time. The bending at any point on the surface segment contributes to the evaluated value leading to slight averaging effects, i.e., singular occurrences of a high surface bending will not necessarily trigger a subdivision of a mostly flat surface piece if the threshold is chosen high enough.

When this effect is not desired, the deviation from the average normal can provide better results. This functional is more susceptible to local spikes, leading to very high subdivision levels around them. However, the subdivision threshold should be modified carefully, as minor changes can drastically increase or reduce the amount of generated viewpoint candidates. Section 7.3 analyzes this behavior in more detail.

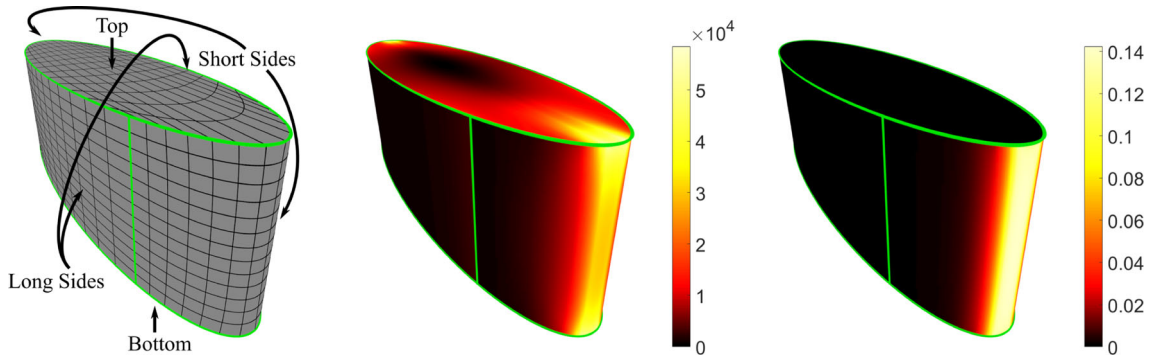
The maximum incidence angle functional is more practically relevant, as it ensures that the entire segment assigned to a viewpoint can be inspected under a certain angle. However, as Fig. 7d shows, the intermediate viewpoint placement evaluation during individual steps can lead to strong fluctuations of the subdivision depth.

Using subdivision by surface area provides regular sampling. It ensures that no segment is larger than the given threshold. It does not highlight any particular properties of the object on its own. However, it is a good supplement for any other criteria that focus on high subdivisions around specific features, e.g., the region of interest.

Table 2 Qualitative overview over different aspects of the presented feature functionals

Functional	Intuition of values	Computational cost	Meeting design objectives	Additivity
Thin plate energy	− −	+++	+	Yes
Curvature	−	− −	++	Yes
Normal deviation	++	−	++	No
Incidence angle	++	−	++	No
Area	+	−	++	Yes
Region of interest	+	−	+++	Yes

The intuition of values describes how easy it is for a human operator to understand the values obtained by evaluating the given functional. Computational cost is important when the system has to be set up interactively or close to real time. How well a functional meets the design objective determines to which extend the results and the user's expectations will align. Fulfilling the additivity property leads to a stable behavior with respect to modifications of the subdivision threshold



(a) Model with isolines **(b)** Thin plate energy **(c)** Curvature

Fig. 11 Analysis of the stretched cylinder. **a** Shows the model with isolines corresponding to the individual knots τ_i of the B-spline surfaces. Planar distortion can be observed on the top surface, caused by forcing a quadrilateral surface into an elliptic shape. As a consequence, the thin plate energy **b** does not vanish. **c** Shows that the curvature values conform with the expected result

In fact, the region-of-interest functional is the most interactive of the presented subdivision criteria. It allows users to highlight important features manually without the need to formally define and implement a measurement. On more complex models, the features of interest and the regions they are occurring in might be detected using a domain specialized tool.

In general, all of the curvature-related functionals lead to a good overall viewpoint candidate placement, ensuring that the object is covered from all angles with a relatively low amount of viewpoints. They automatically summarize the object's geometric complexity into meaningful values freeing the user of the need to specify complex properties manually. The remaining measurements are tailored for specific situations and do not necessarily aim at full coverage of the object in highly bent areas. This can be compensated by combining them with one of the curvature-based criteria.

Table 2 provides a summarized overview over the strengths and weaknesses of the discussed criteria. The following subsections will further discuss individual aspects.

7.1 Curvature computation approach

While the thin plate energy is a widely used functional in variational design to approximate surface bending, it is important to keep in mind that it has been motivated by the physics of deformations. Therefore, it does not accurately model the actual geometric properties of the object. A simple example can be constructed by considering a parametric surface S° with rectangular parameter region $\Omega^\circ = [u_{\min}, u_{\max}] \times [v_{\min}, v_{\max}]$ modeling a flat disk. It is clear that $\kappa_1 = \kappa_2 = 0$ everywhere which means that there is no geometric curvature. Hence, $E_\kappa(S^\circ, \Omega^\circ) = 0$. However, the mapping from a rectangular region to a circular object has to include planar distortions, i.e., second derivatives do not vanish everywhere, implying that $E_{\mathrm{TP}}(S^\circ, \Omega^\circ) \neq 0$.

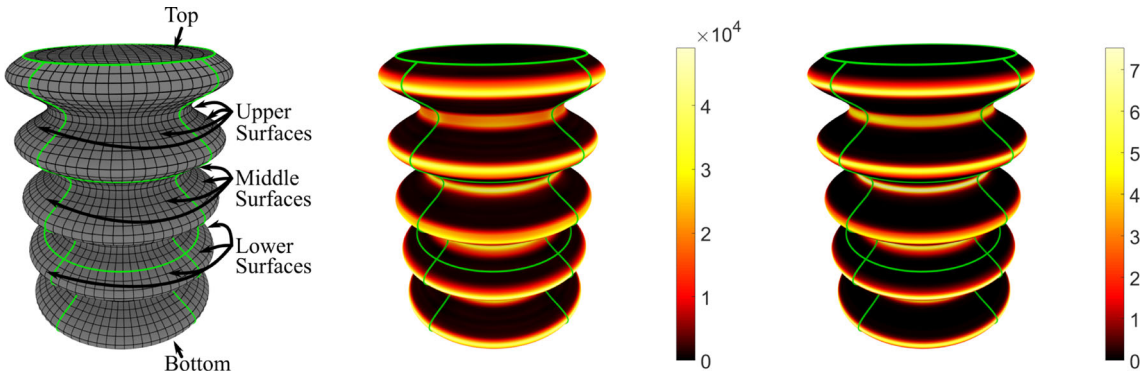
Figure 11b demonstrates this effect. It shows the symmetrical cylinder model consisting of 6 individual B-spline surfaces with a color plot of the thin plate energy integrand

$$\|\partial_{uu} S(u,v)\|^2 + 2\,\|\partial_{uv} S(u,v)\|^2 + \|\partial_{vv} S(u,v)\|^2$$

Table 3 Distribution of feature values on the cylinder model shown in Fig. 11

Functional	Top/bottom (2×) (%)	Long sides (2×) (%)	Short sides (2×) (%)
$E_{\mathrm{TP}}(S)$	18.83	2.95	28.22
$E_{\kappa}(S)$	0	0.41	49.59

While the curvature functional shows the desired distribution, using the thin plate energy will cause a subdivision of the flat surfaces on the top and bottom before considering the slightly curved side surfaces



(a) Model with isolines **(b)** Thin plate energy **(c)** Curvature

Fig. 12 Analysis of the spring model. The isolines (**a**) of the B-splines indicate less distortion in planar regions than for the cylinder in Fig. 11. Although this still causes low nonzero values for the thin plate energy there, the color plot **b** shows that they can be neglected. The distribution is almost identical to the curvature shown in (**c**)

Table 4 Functional values on individual surfaces of the spring model shown in Fig. 12

Functional	Top/Bottom (2×)	Lower surfaces (4×) (%)	Middle surfaces (4×) (%)	Upper surfaces (4×) (%)
$E_{\mathrm{TP}}(S)$	0.27%	8.46	8.16	8.25
$E_{\kappa}(S)$	0	8.71	7.65	8.64

The distribution of values of thin plate energy and curvature mostly coincide, as the low thin plate energy values on the top and bottom surface are almost neglectable

on the surface. Since the top and bottom surfaces have ellipsoidal shapes, their second derivatives do not vanish—especially close to the curved boundaries. At the same time, the surfaces on the side with almost no bending are described by a close to uniformly distributed grid of B-spline control points. Hence, there is only little thin plate energy. Applying one of the presented subdivision approaches leads to a subdivision of the top and bottom before considering these two side surfaces. However, as Table 3 shows, the most energy is located at the highly curved surfaces at the ends, which means that the design objective (i.e., subdivision of highly curved regions) is still met.

The exact curvature measured by $\kappa_1^2 + \kappa_2^2$ is shown in Fig. 11c. Like the thin plate energy, it has a low value on the long side surfaces and a high value on the strongly curved ends. However, it vanishes on both the top and the bottom, which is in alignment with the expected result when aiming for a curvature-based subdivision.

The cylinder model has a relatively simple curvature situation (at least one principal curvature is zero at any point) causing the differences to be distinctively visible. In fact, the value distributions are more aligned when considering a more complex example.

Figure 12 shows the thin plate energy and curvature functional values for the spring model. They are summarized in Table 4 highlighting that there are still planar distortions causing nonzero values of thin plate energy on the flat top and bottom. However, compared to the high values occurring on the sides, they are almost neglectable. A qualitative comparison of the color plots of the integrands of $E_{\mathrm{TP}}(S)$ (Fig. 12b) and $E_{\kappa}(S)$ (Fig. 12c) on the surface leads to the conclusion that both functionals behave similarly for this model.

For a curvature-based subdivision, these comparisons indicate that using $E_{\kappa}(S)$ yields more accurate results as it always exactly meets the design objective. On more complex models, the difference becomes less significant. While it can lead to some additional viewpoint candidates in unintended regions, $E_{\mathrm{TP}}(S)$ still prioritizes highly bent regions leading to the indented subdivision there. However, evaluating $E_{\kappa}(S)$ requires a large number of operations, while $E_{\mathrm{TP}}(S)$

can be computed by simply evaluating Eq. (9). Hence, at the cost of some accuracy, thin plate energy can be used in real time, allowing a user to interactively modify the subdivision threshold. On the other hand, if the viewpoint candidates do not need to be generated interactively and time is not a critical factor, using the curvature functional is preferable.

7.2 Intuition behind feature functional values

An important aspect for the usability of any method is how easy it is for the operator to understand and control the parameters. While the principles underlying the feature functionals can be understood fairly easily, choosing a good termination threshold requires some knowledge of the expected values.

To a user, the integral values evaluated for the thin plate energy or curvature are just numbers with no intuitive meaning. Instead of defining a fixed number as threshold, it is easier to normalize those values and set the threshold to a percentage of the total or average measurement. This is possible since all presented integral measurements can be summed up to a "total" value for the entire object.

The measurements for area and the region of interest are somewhat more intuitive. If the operator has a feeling for the dimensions of the objects, it is possible to define a fixed number of maximum area for the subdivision. However, since those measurements are also integral based, the approach of using an average value also proves viable here.

While a threshold for the maximum angle-based measurements could also be chosen in relation to their average, it is actually more intuitive to set a fixed number for those functionals. The evaluated values are angles, which might already be explicitly determined by external constraints. Even if not, it is fairly intuitive for a user to understand the meaning of a maximum angle when setting up the parameters for the system.

7.3 Stability

To apply the recursive subdivision, a termination threshold needs to be chosen. Unless already given by some external constraints, a system engineer would usually determine this value experimentally, e.g., as described in Sect. 3.2.1. Figure 13 shows examples of this process for selected feature functionals.

Besides the intuition of the expected values, it is also important to be aware of the stability of the used feature functionals. In an optimal setting, small changes to the threshold value only add or remove a low amount of viewpoint candidates. However, in some cases, a small variation in the threshold can make the difference between subdividing a surface several times and no subdivision at all (e.g., Fig. 7c, d).

Such behavior is analyzed by applying the iterative subdivision, i.e., always subdividing the surface with the highest feature value until a certain number of iterations have been done (Algorithm 2). Plotting the highest occurring feature functional value in relation to the number of iterations enables visual analysis of the stability behavior. Figure 14 demonstrates this approach for three individual surfaces. Each surface gets processed with 100 iterations of the subdivision approach. The plots show the behavior of all presented feature functionals except the region of interest, as it is just a restricted variety of the area measurement. Computing the recursive subdivision with a threshold value t corresponds to finding the first intersection between the graph and the horizontal line $y = t$.

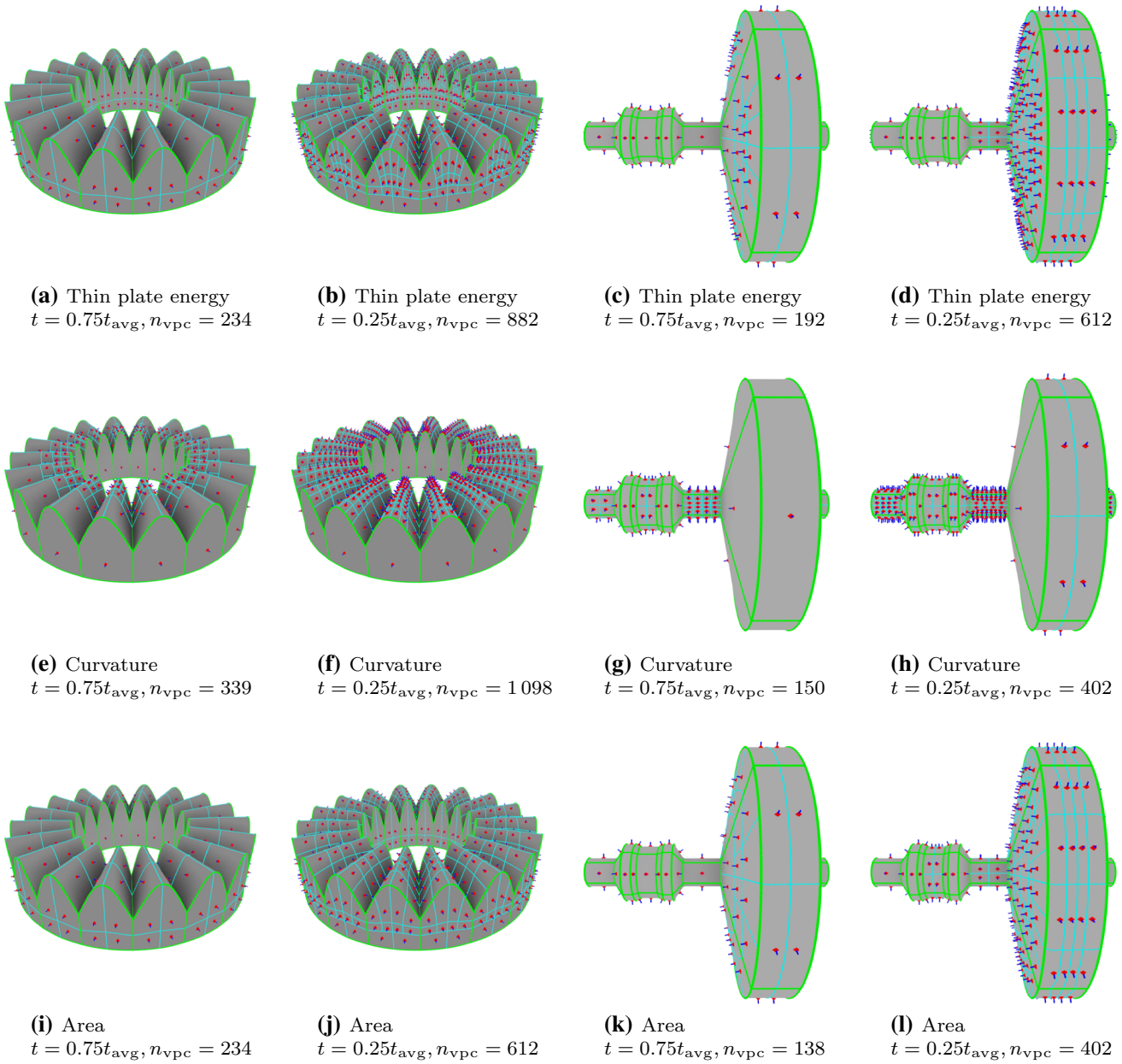
Due to the partition of values when splitting a surface, the integral-based criteria show a monotonous decrease that becomes almost exponential when the surface has a regular geometry. This leads to a stable behavior when choosing a threshold, as well as a fast convergence of the recursive method.

The angle-based functionals measure the maximum deviation of any normal vector from the average normal or a ray originating at the camera, respectively. However, when subdividing the surface, both the average normal and the camera position are recomputed, possibly introducing significant jumps. As a consequence, sudden increases in the evaluated maximum deviation can occur. It is clear that these jumps do not occur anymore when the surface gets subdivided into sufficiently small segments. However, for surfaces with high frequency wiggles (e.g., Fig. 14d), it will take many iterations until this criterion is met. For smoother surfaces, these functionals converge almost with the same rate as the integral-based measurements (Fig. 14a).

7.4 Limitations

The subdivision approach does not explicitly address the total coverage of an object. However, for the presented models this aspect does not pose a problem as the subdivision is always fine enough to ensure that at least one viewpoint is available everywhere. If a guaranteed total coverage or at least coverage of certain parts is desired, a feature functional can be defined measuring the uncovered areas. One way to address this issue is by modeling a viewing frustum through an angular criterion, e.g., by measuring the maximum deviation between camera view direction and view vector to any point on the surface.

Furthermore, the algorithm does not deal with self-occlusions and collisions of viewpoint candidates with the object's geometry. Holes or tight cavities still pose a challenge. Of course, the task of doing visual surface inspection itself is very complex. While it is desired to have a general and adaptable approach suitable for this task, finding a method that is able to process completely arbitrary objects automatically without any intervention or constraints still



(a) Thin plate energy $t = 0.75t_{\text{avg}}, n_{\text{vpc}} = 234$
(b) Thin plate energy $t = 0.25t_{\text{avg}}, n_{\text{vpc}} = 882$
(c) Thin plate energy $t = 0.75t_{\text{avg}}, n_{\text{vpc}} = 192$
(d) Thin plate energy $t = 0.25t_{\text{avg}}, n_{\text{vpc}} = 612$
(e) Curvature $t = 0.75t_{\text{avg}}, n_{\text{vpc}} = 339$
(f) Curvature $t = 0.25t_{\text{avg}}, n_{\text{vpc}} = 1\,098$
(g) Curvature $t = 0.75t_{\text{avg}}, n_{\text{vpc}} = 150$
(h) Curvature $t = 0.25t_{\text{avg}}, n_{\text{vpc}} = 402$
(i) Area $t = 0.75t_{\text{avg}}, n_{\text{vpc}} = 234$
(j) Area $t = 0.25t_{\text{avg}}, n_{\text{vpc}} = 612$
(k) Area $t = 0.75t_{\text{avg}}, n_{\text{vpc}} = 138$
(l) Area $t = 0.25t_{\text{avg}}, n_{\text{vpc}} = 402$

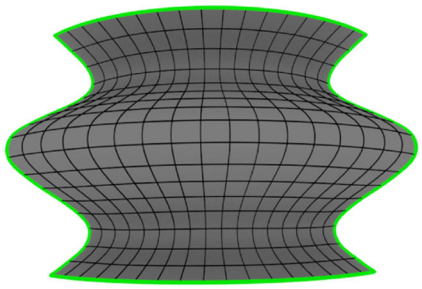
Fig. 13 Subdivision threshold impact. Lowering the subdivision threshold increases sampling density in areas of interest, depending on the feature functional. By adjusting the threshold, an operator can interactively ensure that the object is sufficiently covered, keeping total number of viewpoint candidates low

requires a lot of further research. However, the presented method is designed in a way that allows various modifications to tackle this problem, e.g., by adapting the way the pivot points are chosen or the direction in which the camera is placed. Additional post-processing steps can be applied to correct the positioning of invalid viewpoint candidates.

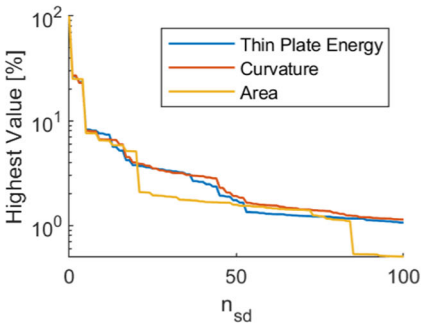
8 Conclusions and future work

The presented method provides an adaptive and intuitive solution for object space exploration by generating desirable viewpoint candidates in sufficient numbers. It allows skilled system engineers to factor in expert knowledge, while at the same time producing viable results when used with the default settings. The experiments have shown that the thin plate energy functional leads to well-distributed viewpoint candidates requiring little computational effort.

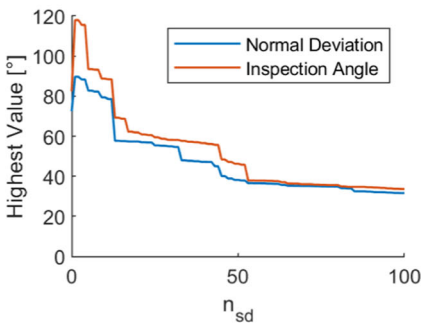
As the results show, objects can be fully covered with a small number viewpoint candidates. Thus, the subsequent NP-hard problem of selecting optimal viewpoints has to be solved only for a small input. The subdivision threshold, and with it the density of the surface sampling, can be chosen



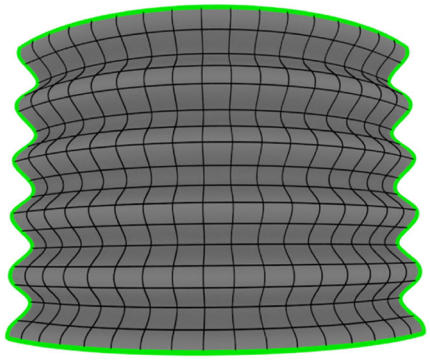
(a) Surface with isolines



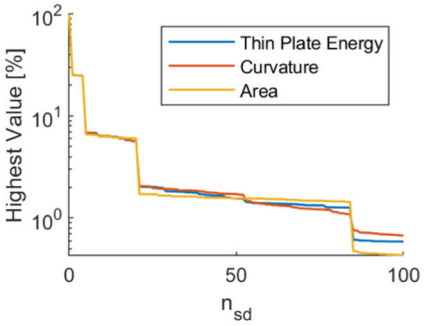
(b) Integral Based Criteria



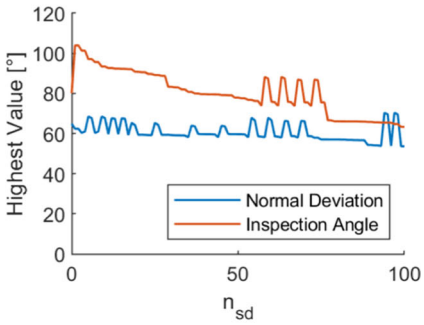
(c) Max Angle Based Criteria



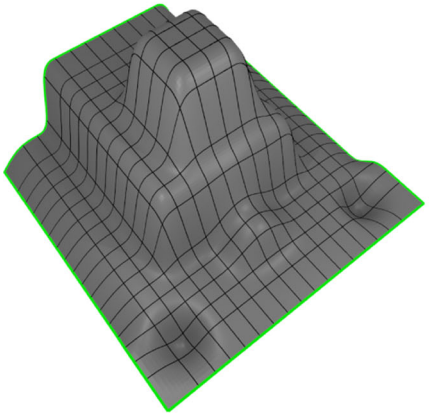
(d) Surface with isolines



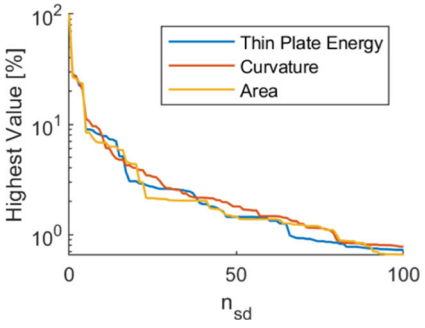
(e) Integral Based Criteria



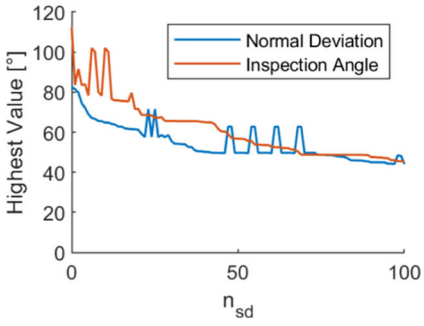
(f) Max Angle Based Criteria



(g) Surface with isolines



(h) Integral Based Criteria



(i) Max Angle Based Criteria

Fig. 14 Analysis of the stability and convergence behavior of the presented functionals. Three selected surfaces (**a**, **d**, **g**) of different geometric complexities are considered. For each surface, 100 subdivision steps are performed. Each plot shows the highest measured value on the y-axis with respect to the number of performed subdivisions n_{sd} on the x-axis. The integral-based values (**b**, **e**, **h**) are given as percentage of the total value measured for the entire surface, as it always equals the sum of all segments. Since the decrease is close to exponential, a logarithmic scale is used. Angle-driven functionals (**c**, **f**, **j**) are given as absolute values. Correspondence to the recursive subdivision can be obtained by finding the first intersection between the plotted graphs and a vertical line representing the subdivision threshold $y = t$

interactively. By pre-computing the subdivision to a certain depth, the operator can modify the threshold and evaluate the impact on the results in real time.

The use of B-splines makes the approach independent of the resolution of a discrete representation, as a single B-spline surface can represent surface triangulations of various resolutions. The analytic and continuously differentiable B-spline representation allows for the computation of various measurements. A user can easily factor in application-specific constraints or define new features to be focused on during inspection. Additionally, the B-spline construction, based on least-squares approximation and a smoothing term, can compensate for small amounts of noise in the discrete input data.

Future extensions of this method will focus on optimizing camera positions in order to consider additional properties like material behavior or light source placement. It is also worth investigating how the subdivision itself can become more adaptive, e.g., to highlight interesting regions with fewer overall subdivision steps. Ultimately, the goal is to have an efficient and purely virtual system that sets up the physical inspection system. This will be of substantial benefit in agile production environments, where a high degree of automation is desired.

The overall pipeline for setting up inspection systems has existed for years and in many variations. Despite the desperate need to do so, full automation for arbitrary objects has not yet been achieved. While the work presented here addresses one singular aspect of the inspection pipeline, it provides researchers and engineers with a powerful and adaptable solution, with the potential for further innovations. Continuous development of this approach will lead to smart inspection systems capable of adapting to new products within minutes, making production and inspection systems truly agile.

Acknowledgements This research was funded by the Fraunhofer High Performance Center for Simulation- and Software-Based Innovation and supported by the German Research Foundation (DFG) within the IRTG 2057 "Physical Modeling for Virtual Manufacturing Systems and Processes," as well as a grant by the Deutsch-Französische Hochschule (DFH).

Funding Open Access funding provided by Projekt DEAL.

References

1. Atkinson, K.E.: An Introduction to Numerical Analysis. Wiley, Hoboken (2008)
2. Beyerer, J., León, F.P., Frese, C.: Machine Vision: Automated Visual Inspection: Theory, Practice and Applications. Springer, Berlin (2015)
3. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
4. Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G.: MeshLab: an open-source mesh processing tool. In: Scarano, V., Chiara, R.D., Erra, U. (eds.) Eurographics Italian Chapter Conference. The Eurographics Association (2008). https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136
5. Cowan, C.K., Kovesi, P.D.: Automatic sensor placement from vision task requirements. IEEE Trans. Pattern Anal. Mach. Intell. **10**(3), 407–416 (1988). https://doi.org/10.1109/34.3905
6. Do Carmo, M.P.: Differential Geometry of Curves and Surfaces: Revised and Updated, 2nd edn. Courier Dover Publications, Mineola (2016)
7. Eck, M., Hoppe, H.: Automatic reconstruction of B-spline surfaces of arbitrary topological type. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 325–334. ACM (1996)
8. Englot, B.J.: Sampling-based coverage path planning for complex 3d structures. Ph.D. thesis, Massachusetts Institute of Technology (2012)
9. Farin, G.: Curves and Surfaces for CAGD: A Practical Guide, 5th edn. Morgan Kaufmann Publishers, Burlington (2002)
10. Fraunhofer ITWM: Toolip tool for image processing (2019). https://www.itwm.fraunhofer.de/en/departments/bv/products-and-services/toolip.html. Online Accessed 26 Feb 2019
11. Gronle, M., Osten, W.: View and sensor planning for multi-sensor surface inspection. Surf. Topogr. Metrol. Prop. (2016). https://doi.org/10.1088/2051-672X/4/2/024009
12. Hagen, H., Schulze, G.: Automatic smoothing with geometric surface patches. Comput. Aided Geom. Des. **4**(3), 231–235 (1987)
13. Huang, J., Zhou, Y., Niessner, M., Shewchuk, J.R., Guibas, L.J.: Quadriflow: a scalable and robust method for quadrangulation. In: Computer Graphics Forum, vol. 37, pp. 147–160. Wiley Online Library (2018)
14. Jing, W.: Coverage planning for robotic vision applications in complex 3d environment. Ph.D. thesis, Carnegie Mellon University (2017)
15. Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: Open source scientific tools for Python (2001). http://www.scipy.org/. Online Accessed 26 Feb 2019
16. Mavrinac, A., Chen, X., Alarcon-Herrera, J.L.: Semiautomatic model-based view planning for active triangulation 3-d inspection systems. IEEE/ASME Trans. Mechatron. **20**(2), 799–811 (2015). https://doi.org/10.1109/TMECH.2014.2318729
17. Mohammadikaji, M.: Simulation-based planning of machine vision inspection systems with an application to laser triangulation. Ph.D. thesis, Karlsruhe Institute of Technology (2019)
18. Nouanesengsy, B., Woodring, J., Patchett, J., Myers, K., Ahrens, J.: Adr visualization: a generalized framework for ranking large-scale scientific data using analysis-driven refinement. In: 2014 IEEE 4th symposium on large data analysis and visualization (ldav), pp. 43–50. IEEE (2014)
19. Prieto, F., Lepage, R., Boulanger, P., Redarce, T.: A CAD-based 3d data acquisition strategy for inspection. Mach. Vis. Appl. **15**, 76–91 (2003). https://doi.org/10.1007/s00138-003-0131-4
20. Sakane, S., Niepold, R., Sato, T., Shirai, Y.: Illumination setup planning for a hand–eye system based on an environmental

model. Adv. Robot. **6**(4), 461–482 (1991). https://doi.org/10.1163/156855392X00295
21. Sakane, S., Sato, T.: Automatic planning of light source and camera placement for an active photometric stereo system. In: Proceedings. 1991 IEEE International Conference on Robotics and Automation, vol. 2, pp. 1080–1087 (1991). https://doi.org/10.1109/ROBOT.1991.131737
22. Scott, W.R.: Model-based view planning. Mach. Vis. Appl. **20**, 47–69 (2009). https://doi.org/10.1007/s00138-007-0110-2
23. Sheng, W., Xi, N., Song, M., Chen, Y., Rankin, J.S.: Automated CAD-guided automobile part dimensional inspection. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), vol. 2, pp. 1157–1162. IEEE (2000)
24. Sheng, W., Xi, N., Tan, J., Song, M., Chen, Y.: Viewpoint reduction in vision sensor planning for dimensional inspection. In: IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003, vol. 1, pp. 249–254. IEEE (2003)
25. Slavk, P.: A tight analysis of the greedy algorithm for set cover. J. Algorithms **25**(2), 237–254 (1997). https://doi.org/10.1006/jagm.1997.0887
26. Tarabanis, K.A., Allen, P.K., Tsai, R.Y.: A survey of sensor planning in computer vision. IEEE Trans. Robot. Autom. **11**(1), 86–104 (1995). https://doi.org/10.1109/70.345940
27. Tarabanis, K.A., Tsai, R.Y., Allen, P.K.: The MVP sensor planning system for robotic vision tasks. IEEE Trans. Robot. Autom. **11**(1), 72–85 (1995). https://doi.org/10.1109/70.345939
28. Tarbox, G.H., Gottschlich, S.N.: Planning for complete sensor coverage in inspection. Comput. Vis. Image Understand. **65**(1), 84–111 (1995). https://doi.org/10.1006/cviu.1995.1007
29. The CGAL Project: CGAL User and Reference Manual, 4.14 edn. CGAL Editorial Board (2019). https://doc.cgal.org/4.14/Manual/packages.html

Dennis Mosbach is a researcher at the image processing department of Fraunhofer ITWM in Kaiserslautern, Germany. His main research areas are geometric modeling, computer graphics, and 3D image processing. He studied applied computer science at the University of Kaiserslautern, Germany.

Petra Gospodnetić has received her master's degree from Faculty of Electrical Engineering and Computing, University of Zagreb in 2016. She is currently a PhD candidate working on a close collaboration with Technical University of Kaiserslautern and Fraunhofer ITWM. Petra is working on development of visual inspection systems and performance optimization. Her focus lies on inspection planning automation through serialization of the involved processes and virtualization of the everyday planning procedures performed by an expert. In order to make it possible, she combines multiple fields such as computer vision, computer graphics, robotics and machine learning.

Markus Rauhut Dipl. Inform. Markus Rauhut studied computer science and mathematics at the Technical University of Kaiserslautern. After working as a software developer in industry for a few years, he joined the Fraunhofer ITWM in 2001. Since 2014, he is head of the department Image Processing at Fraunhofer ITWM. He has many years of experience in processing and managing projects at the interface between research and application. His research interests are in the fields of algorithms for image processing and the development of complete solutions for image processing systems.

Bernd Hamann is a computer scientist serving at the University of California, Davis. His main teaching and scholarly interests are visual data analysis, geometric computing and image processing. He studied computer science and mathematics at the Technical University of Braunschweig, Germany, and Arizona State University.

Prof. Dr. Hans Hagen is a computer science professor at University of Kaiserslautern and an adjunct professor at University of California, Davis. He received a Bachelor's degree in computer science , a Master degree in mathematics from the University of Freiburg and a Ph.D. in mathematics (geometry) from the University of Dortmund. His main research interests are scientific visualization and geometric modeling. He is a member of the IEEE Visualization Academy of Science, and he got the IEEE Visualization Career Award, the ACM Solid Modeling Pioneer Award and the John Gregory Memorial Award among others.