Surface Inspection Planning Using 3D Visualization

Petra Gospodnetic*, Markus Rauhut*, Hans Hagen†

*Image Processing Department, Fraunhofer ITWM, Kaiserslautern, Germany
E-mail: petra.gospodnetic@itwm.fraunhofer.de, markus.rauhut@itwm.fraunhofer.de

†Computer Graphics and HCI Group, University of Kaiserslautern, Kaiserslautern, Germany
E-mail: hagen@cs.uni-kl.de

Abstract—State of the art surface inspection planning requires an expert approach with a lot of trial and error. Because of the lack of available aiding tools, an engineer building inspection systems must rely heavily on his or her experience. In this work we proposed an interactive 3D visualization tool to help an engineer determine good viewpoints. It can be used both as a standalone tool for manual viewpoint placement or as an interface to the inspection planning algorithms giving the engineer a possibility to evaluate and modify automatically generated plans.

Index Terms—visual surface inspection, inspection planning, 3D visualization

I. INTRODUCTION

Modern production lines frequently use automatic visual inspection systems as a way of ensuring product quality. Because the products are diverse and have different requirements, inspection systems are often custom designed. Design of such systems is an iterative process, relying heavily on the experience of the engineer conducting it. First step is to build a prototype system, representing a proof of concept. Based on experience, the prototype development time varies greatly, depending on the object to be inspected and inspection requirements. It may take anywhere from 1/10 (for simple cases), up to 1/2 of the overall development time, which, itself, can last up to 3 years for very complicated products.

The prototype design consists of image acquisition and algorithm design steps, where the image acquisition design defines the physical arrangement of the hardware components such as camera and illumination. For that purpose an engineer must decide which devices should be used (e.g. camera, optics, illumination) and how should they be positioned — relative both to each other and the object (acquisition setup). Proper choice and positioning of the devices is crucial for ensuring that the acquired image shows the best possible differentiation between the defects and the object surface. Within the subsequent step of algorithm design, various image processing algorithms are tested and developed as necessary.

The image acquisition has a big impact on the overall quality of an inspection system, but is still determined heuristically, requiring the engineer to have a lot of experience and conduct a lengthy trial-and-error testing process. During the process, a number of acquisition setups required to inspect the object in its entirety is determined. The length of the process increases significantly with the rise of geometrical

complexity of the object to be inspected, due to variation in surface orientation and occurrence of self occlusion. Therefore, assuming that complete surface needs to be inspected, a geometrically complex object is likely to require a lot more separate acquisition setups than a simple one. Finally, the chosen acquisition setups may be difficult to evaluate in terms of coverage and redundancy, i.e. which parts of the object are covered, at which angle and how much do they overlap between different setups. The problem described so far is more commonly known as the inspection planning problem and has caused a significant amount of research in the area of inspection planning automation over the past decades. Due to its complexity however, the problem has not been solved yet.

We suggest that an interactive 3D visualization tool, capable of replicating at least parts of the physical planning process and enabling evaluation of different planning setups, would make the inspection planning problem more comprehensible. In this work we present such tool, aimed to provide support to an engineer planning a visual surface inspection by enabling arbitrary camera viewpoint placement and real-time camera coverage estimation. Furthermore, the tool can be used as an intermediary step in an automated surface inspection planning pipeline, enabling the engineer to restrict the planning to specific regions, as well as to both evaluate and modify the planning results.

II. RELATED WORK

Inspection planning is an open problem within the research community focusing on inspection systems. So far it has been observed from an optimization point of view, researching various approaches to deal with the complexity of the planning problem. [5], [6], [10], [17], [18] focused onto finding a single optimal solution on how to inspect an object, given its 3D model. In simple terms, how should the acquisition system be designed, relative to the object, in order to inspect it completely. Ideally, one such solution contains two sets of variables. A set of primary variables, describing the spatial relations of the camera, object and the illumination within the acquisition system, and a set of secondary variables, such as camera lens, exposure time, sensor resolution, illumination brightness and wavelength, etc. [5], [6], [17] and [18] offered approaches to determine viable camera placement. However, by providing both camera and illumination placement, Mohammadkaji et al. [10] were the only ones to succeeded in

producing an inspection plan containing all primary variables. Their method, however, comes with a restriction of high computational cost and is hard to generalize to a wide variety of objects.

All of the above mentioned works assumed that, beside the input data, no further human interaction is required. Also, all of them noted that an increase of the geometrical complexity of an object makes the optimization complex, given that the problem at hand is a 3D generalization of the NP-hard art gallery problem [9].

Mosbach et al. [11], took a different approach and focused on the object space exploration problem. Instead of trying to find the optimal positions for the camera placement, they decided to tackle the problem of geometrical complexity by sampling the surface with various density. Their approach uses a set of so-called feature functionals to analytically measure geometrical features of an object and use them to guide the sampling process (e.g. higher sampling in the curved areas and lower in the flat areas). The produced points can further be used as camera or illumination viewpoint candidates.

Reiner [15] recognized the value of computer graphics for the purpose of machine vision prototyping. He suggested to use the physically-based rendering technique as a mean to evaluate the image acquisition results, without physically building the setup. The work focused on the impact and use of illumination profiles on image acquisition, therefore we consider his work to be focused on the evaluation aspect of the inspection planning. Irgenfried et al. [7], [8] explored the possibility of using rendered images as a mean to develop a data set for testing machine vision algorithms for edge detection. The process was later made part of a toolset for computer aided inspection planning. It offers a possibility to annotate CAD files using a proprietary tool, write a configuration file describing the system configuration in an XML based file format and visualize the 3D model of the object overlayed with the results of automatic inspection planning. It is also possible to obtain physically-based rendered images simulating the expected sensing result. While the CAD annotation is interactive, the 3D model visualization is used solely for the purpose of result evaluation.

Commercial quality management solutions such as [2], [13], [16] focus on inspection process flow and task management, rather on an actual inspection setup and data acquisition. So far no inspection planning solution is widely available, which would aid the engineer in inspection planning tasks. For that purpose, in this work, we suggest a single solution using interactive 3D visualization. 3D graphics software Blender [4] and processing tool MeshLab [3] inspired this work to embrace a similar model centered workflow, as well as face selection capabilities. Both tools employ a plethora of different modelling and processing options, many of which are redundant for planning purposes. The tools have a 3D model as an end goal, while in case of inspection planning, a 3D model is a static central piece in relation to which inspection setup is designed. Therefore, we will focus on model selection tools.

MeshLab [3] offers many selection options for both vertices and faces, of which we find interesting region, view angle and boundary based selection. Region selection selects all the elements within a cursor drawn rectangle along the z-buffer. View angle based selection uses a mesh camera or an arbitrary viewpoint to select all faces whose normal close a predefined angle with the viewpoint viewing direction. Neither region or angle based selection has an option to take visibility of the face or self occlusion into the account. Finally, boundary based selection only works for non-closed models and 2D models and selects vertices and faces creating the topological boundary. The method, however, is not applicable for selection of faces along a sharp edge of a mesh as it is not considered a boundary. Blender [4] also allows rectangular (box) selection, but in contrast to MeshLab, it can take the z-buffer into account and provides a possibility to select only visible faces or vertices. Beside boundary selection, it is also possible to select at once all the sharp edges where face normals close an angle smaller than a predefined angle. Any made selection can be further expanded by selecting co-planar faces or those with similar normal direction. Neither Blender or MeshLab offer a straightforward possibility to place a viewpoint relative to the object, evaluate its coverage and export it for future use.

III. APPLICATION REQUIREMENTS

During the inspection planning process an engineer is expected to perform a lot of physical work while remounting and testing various camera and illumination devices and configuration. It is the responsibility of the engineer to ensure that all surfaces of the object are adequately inspected (covered). For the purpose of this work we will focus on the camera placement and coverage, assuming that the appropriate illumination will be determined subsequently by an engineer.

As explained in Section II, geometrical complexity of an object influences the planning process significantly. In the case of an object with a low geometrical complexity, camera viewpoints which will yield sufficient object coverage may not be hard to determine heuristically. However, due to the frequent occurrence of self occlusion and high variation of angles under which a surface is observed, within a single image, choosing good viewpoints becomes non-trivial. Furthermore, complex objects may require many more viewpoints in order to obtain the full coverage of an object. Currently no solution offers a possibility to determine the amount of object surface covered by each viewpoint. Therefore, an engineer must physically reposition, fix and test different viewpoints, as well as introduce redundancy in order to satisfy inspection requirements.

Given the above stated problems, a planning tool should enable good camera placement estimation and coverage evaluation, regardless of the geometrical complexity of the inspected object. The tool should be designed specifically for inspection planning purposes and interaction with it tool should resemble the physical planning process. It should be intuitive, provide real-time feedback, be applicable to any object (given its 3D model) and have a possibility to export viewpoint plan and system configuration in a standardized format. Since the

Fig. 1: Viewpoint visualization for the gear object, containing 686 viewpoint candidates and 135 optimal viewpoints. In *a)* only a viewpoint candidate set is visualized with full opacity. In *b)*, the viewpoint candidate set opacity is reduced and the optimal viewpoints are overlaid. In *c)*, the optimal viewpoint set was modified by deleting viewpoints, and a single viewpoint is selected. The selected viewpoint is colored in red, and reveals the coverage impact the viewpoint has on the gear object.

automatic inspection planning approaches so far concentrated on producing inspection plans, but not the possibility to modify them by an experienced engineer, the tool should bridge the gap and provide a possibility to load an inspection plan generated by another inspection planning procedure and make modifications to it.

This tool does not aim to suggest or determine good viewpoints, rather provide the engineer with a possibility to do so.

IV. PLANNER OVERVIEW

The inspection planning tool is aimed to help the engineer who is building inspection systems to determine appropriate camera positions in order to cover the complete object. The illumination placement is not considered within this work and is assumed to be ideal, i.e. each observed point can be illuminated to be visible by the camera. Position and orientation of the camera is defined as a viewpoint, a 6D vector describing a 3D point and its 3D orientation, relative to the origin of the 3D model.

Focus of the tool lies on the object space exploration, annotation of 3D models, as well as evaluation and modification of both viewpoint candidates and calculated optimal viewpoints. In contrast to [8] and [15], it is not a realistic simulation of the sensed image, but a visualization tool providing insight into the spatial distribution of the viewpoints and their impact on the object in terms of coverage.

Since production lines are following the trend of becoming easily customizable and capable of producing various products on the same line, an inspection plan may need to be developed for more than one object at the time. Therefore, it is possible to upload several objects and work on them in parallel.

A. 3D Model Exploration

When an engineer considers a potential camera viewpoint, it is always relative to the object. Therefore, if possible, the engineer will pick up the object and manually observe it from

all sides while rotating it. The action can be identified as object space exploration, enabling consideration of different placement possibilities. Interactive 3D model visualization translates the same action into the digital world, enabling the engineer to repeat the same thing, without actually having to do it physically.

The chosen 3D model is visualized using Phong material [12] and illuminated by static scene lights, used only for model exploration purposes and therefore not affecting any planning results. Model handling functionalities such as rotation, translation and scaling eliminate any physical handling restrictions such as the size and the weight of the object (e.g. large and heavy mechanical parts or very small clock mechanism components). Since digital inspection planning and object exploration is computed per-face, coverage highlighting is done per-face as well. Therefore, to provide the user with better understanding of the results, an option to examine the tessellation of the model by enabling the *wireframe view* is available.

B. Viewpoints Placement and Evaluation

When deciding which viewpoints should be used for the inspection, an engineer must approximate how will they impact the object, i.e. which areas will be covered by a particular viewpoint. When faced with an object, there is an unlimited number of possibilities where to place the camera and how to orient it. Normally the problem is solved by choosing the solution based on the experience. However, an algorithm performing the automatic inspection planning can not use the experience and must therefore discretize the solution space and choose its solution based on the sampled points in space which are referred to as viewpoint candidates. For every viewpoint candidate, the precomputed coverage is passed to the visualization tool, in order for the tool to be able to visualize the same coverage used when choosing optimal viewpoints, as described in [11].

Fig. 2: Selection of multiple viewpoints will reveal the total area covered (red), as well as the area in which their coverage overlaps (green).

1) *Visualization*: the planning tool uses cylinders to visualize viewpoints. Side of the cylinder which is further from the mesh represents a point in space where the camera should be placed and the body of the cylinder denotes the direction of the camera's optical axis. The cylinders are set to the length of the provided focusing distance in order to give the viewer a straightforward notion of the part of the object being observed, without any interaction.

It is possible to visualize both viewpoint candidates and one or more optimal viewpoint sets, given that they may be generated using different parameters. In addition, the user can modify viewpoint orientation, delete them or manually add new ones. For increased understanding of the correspondence, viewpoint sets can be overlaid. Overlaying several sets of viewpoints may crowd the space around the object, especially given the fact that viewpoint candidate sets may contain up to several thousands of viewpoints (e.g. [6], [11], [18]). Therefore, for each set of viewpoints it is possible to change its color, thickness and opacity in order to achieve maximal distinction between them, as can be seen from Figure 1.

2) *Selection and Coverage*: Each of the viewpoints has a unique index, which can be used to select the viewpoint and its coverage on the object directly through the menu. Alternatively, hovering the mouse over a viewpoint will highlight both the viewpoint in focus and the corresponding coverage. Selecting it will display its index together with all the data it contains (e.g. position and orientation, coverage impact, etc.). Selection will also keep the viewpoint and the coverage highlighted so that it can be compared to any other viewpoints. Hovering every subsequent viewpoint will then, in addition to highlighting the coverage, reveal if there is any overlap

between the viewpoints selected so far, and the hovered one, as can be seen in Figure 2. Finally the user can enable the full coverage highlight, emphasizing which portion of the object is covered if all of the viewpoints from the set are used.

3) *Modification and System Configuration*: For the first time, the user has the possibility to intervene with the planning results. The initial set of viewpoint candidates used by the inspection planning algorithm, as well as any of the resulting sets of optimal viewpoints, can be adapted. Furthermore, any parameters used for the coverage computation (e.g. camera resolution, pixel size, focal length, depth of field interval and maximal incidence angle), or planning parameters (e.g. required coverage threshold) are also exposed to the user and can be changed.

New viewpoints can be added by selecting a so-called pivot point on the model. The normal of the facet containing the point is further used as a reference vector. By default, a new viewpoint is generated along the reference vector, without any change, at the focusing distance. However, the orientation of the reference vector can also be adapted by altering the parameter representing the angle the viewpoint should close with the surface. Finally, the camera is set to look at the object along the negative direction of the reference vector. Orientation and angle the viewpoint closes with the surface can also be modified after the viewpoint is created through selection and mouse movement. The viewpoint will then be rotated around the pivot point. For every added viewpoint a well known rasterization technique is employed to compute newly covered surface and update the overall coverage.

While adding viewpoints, using a preview option, the user can have a visual feedback on the coverage impact a viewpoint would have if placed on the point cursor is pointing to. It will show viewpoint coverage as well as overlap with surface covered so far by other viewpoints.

By altering the provided parameters, it is possible to generate configuration files for the external processes, such as viewpoint candidate creation, coverage computation or viewpoint optimization. Additionally, the user can export the modified viewpoint sets or a completely new plan if needed.

C. Inspection Analytic

In order to provide the user with all the planning information available, various analytic information is shown. *3D model information* displays the format of the model and the number of faces and vertices in it. *Viewpoint candidate information* displays the total number of viewpoint candidates, feature functional used to generate them, together with the used threshold. *Coverage information* displays the time needed to compute the coverage of the entier set and the information about the camera parameters used in the process, i.e. sensor resolution, pixel size, focal length, depth of field and maximal angle of incidence. *Viewpoint optimization* displays the name of the used optimization algorithm, time required to do the optimization and number of optimal viewpoints.

Fig. 3: Oriented plane filling requires from the user only to (a) paint a small sample of the (b) surface which should be selected (right).

D. Selection of Inspection Regions

Together with the clients ordering the inspection system and upon careful examination of the object, the engineer might have the need to use different camera parameters for various parts of the object. For that purpose regions of interest can be defined and selected either manually, or using selection filters. Manual annotation can be especially useful for free form surfaces. One object can have multiple separated regions of interest. A region of interest contains a set of facets, represented by indices, which correspond to facet order in the 3D model file format. That way, the region of interest can be passed to other planning programs, under the assumption that the same 3D model file will be used. Aside of the facet indices, every region can be assigned specific camera parameters which should be used for its inspection. The parameters depend on the parameters defined by the external process used for the coverage calculation (see Section V) and can be expanded if necessary.

1) *Manual selection*: using a brush, the user can select or deselect facets of the object in a painting manner. For every click, or move of the mouse while pressing the button, a facet is selected. It is done by casting a ray from the mouse screen coordinates into the scene and finding the intersecting facet on the mesh. Size of the brush can be adapted and is defined as k nearest neighbors, relative to the intersected facet.

2) *Selection modifiers*: the nearest neighbor search used for region painting is based only on the distance between facet centroids. Therefore, the closest facets must not necessarily be connected, or even lie in the same plane. While it is not likely to cause problems when selecting surfaces with small normal variation, for complex surfaces it may cause an unwanted effect of over-selection. The effect is particularly obvious in areas around sharp edges of the mesh. Beside the over-selection, manual annotation can be a tiring process to do precisely. It becomes more obvious as the complexity of the object rises. For that purpose, manual selection can be used in combination with selection modifiers, making it easier for the user to achieve the desired result. Once a region is selected, selection modifiers can be applied to expand the selection or

Algorithm 1 Axis aligned bounding box search

```

function ISBBOXWITHINNODE(node)
   $v \leftarrow \text{CLOSESTBBOXVERTEX}(\text{node.point}, B)$ 
  if INBBOX( $v$ ,  $\text{node.BBox}$ ) then
    return true
  else
    return false
  end if
end function

function BBOXSEARCH(treeNode)
   $n \leftarrow \text{treeNode}$ 
   $p \leftarrow \text{treeNode.point}$ 
  if ISLEAF( $n$ ) then
    return
  end if
  if INBBOX( $p$ ,  $B$ ) then
    SAVE( $n$ )
  end if

  BBOXSEARCH( $n.\text{leftChild}$ )
  if INBBOX( $p$ ,  $B$ ) or ISBBOXWITHINNODE( $n$ ) then
    BBOXSEARCH( $n.\text{rightChild}$ )
  end if
  return savedNodes
end function

```

introduce further restrictions.

Average orientation filter can be found useful in cases where the over-selection happened while the user was selecting a surface with low normal variation. First an average normal of the selected area is calculated. Then, an angle between the average normal and normal of every selected facet is measured. If the angle is greater than a predefined threshold, the facet is considered to be over-selected and is rejected from the selection, as can be seen in Figure 5. The threshold is exposed to the user and can be adapted as needed. While such approach has the benefit of not being bound to the axis, it also poses the requirement that the over-selection takes only minor portion of the overall selected region. Otherwise, it influences the average normal too much and will cause filtering to produce unwanted results.

Oriented plane filling is an alternative approach to orientation filtering. The user must only paint a small portion of the region which should be selected. The normal of the last selected facet is used as an orientation reference and its centroid as a reference point-in-plane. First, all the facets with the same orientation need to be retrieved. Once the facets containing the same direction are retrieved, they are tested whether they are lying in a reference plane or not. It is done by creating a vector connecting the centroid of the tested facet and the reference point-in-plane, and calculating the dot product between the two. If the result is close to 0, the tested facet is added to the selection, as can be seen in Figure 3.

Edge filtering is useful for cases when the user wishes to mark edges as regions of interest. Such need is quite usual, as many product are considered to be defect only if the defect appears along the edge of the object. If done solely using painting, it is nearly impossible to make a selection with a

Fig. 4: 3D models used in the pipeline. (a) stretched cylinder (21,800 triangles), (b) spring (88,468 triangles), (c) tanglecube (82,672) and (d) gear (169,472 triangles)

constant distance from the edge. To make the process easier, the user is asked only to loosely select the edge, and then apply the edge filter (Figure 7). For every facet of the selection, the filter will retrieve an arbitrary number of its closest neighbors and select only the ones whose normals are perpendicular to the tested facet. The number of closest neighbors is exposed to the user and can be adapted as necessary.

3) *Automated selection*: the user may want to completely avoid the manual selection step, or make the selection restricted in terms of position or orientation.

Axis aligned bounding box allows the user to select a portion of the object by providing an interval along the each of the three axes. Such functionality becomes particularly useful when the object must be split into axis-aligned regions, as can be seen in Figure 8.

Orientation based selection makes it possible to select only the surfaces of a particular orientation, by providing a reference orientation vector. Just like for the average orientation filter, the user can use the threshold angle to control how much deviation from the reference angle is allowed, as can be seen on 6.

E. Supporting Data Structures

In order to perform selection by raycasting and make facet filtering possible, intensive search is often needed and is computationally expensive. Since the interface requires real-time response, additional supporting data structures are needed to make it possible, i.e. k-D trees [14]. For raytracing, we rely on the structure built by the used renderer [19]. However, for the purpose of the k nearest neighbors search, orientation filtering and bounding box restriction, additional trees are needed. We have used and modified a fast k-D tree implementation by [20]. Two additional trees were needed, one grouping the facet centroids, and the other grouping the facet normals. Beside building additional trees, we introduced a possibility to traverse the tree and return all the points within a given bounding box $B = x_1, x_2, y_1, y_2, z_1, z_2$, where x_1, y_1 and z_1 denote the minimum of the interval and x_2, y_2 and z_2 the maximum of the interval on the corresponding axis. The functionality is described by Algorithm 1.

F. Viewpoint Description Standard

Viewpoints are stored in a JSON file, described by a 3D point, two vectors and an unique index. The point places the viewpoint relative to the origin of the object, while two

TABLE I: Loading times

	Cylinder	Spring	Tanglecube	Gear
Overall loading [s]	0.442	2.32	3.45	5.5
k-D tree creation [s]	0.292	1.94	2.35	5.34

orthogonal vectors describe its rotation using *viewing direction* and camera *up vector*. Furthermore, the file must contain camera information used to compute the coverage, i.e. sensor resolution, pixel size, focal length, depth of field interval, focusing distance and maximum allowed incidence angle. The same format is used both for import and export of the inspection plan. Coverage of each viewpoint is stored in a separate JSON file, where every viewpoint index is linked to a list of face indices covered by that viewpoint.

V. RESULTS

The presented 3D visualization tool for surface inspection planning focuses on inspection of a single object and has therefore been tested on four different 3D models, shown in Figure 4. All of the models are synthetic and show different challenges an engineer might face. The gear is made to resemble an object which is frequently produced. The geometrical complexity varies between the objects, being the lowest for the stretched cylinder and the highest for the tanglecube which introduces both topological challenges, as well as free form surface.

Use of the tool for the purpose of evaluation and modification of automatically generated plans was tested on viewpoints generated as described in [11]. Each of the objects have a set of both viewpoint candidates and optimal viewpoints (Figure 1). The tool itself is, however, completely independent of the viewpoint set generation and can be used to visualize any kind of viewpoints, as long as they can be described using format introduced in Section IV-F.

Coverage computation was performed by a separate raytracing process in case of viewpoints generated by an external planning process and using rasterization in case of manually created viewpoints. A pinhole camera model was used in both cases. Sensor size was $2\,050 \times 2\,448$ px, with a pixel size of 3.45×10^{-3} mm. The focal length was set to 12 mm and the focusing distance was measured to be 130 mm. Acceptable depth of field interval was set to be between 125 and 137 mm and the maximum allowed incidence angle was set to be 60 deg.

The tool was implemented as a web application using AngularJS [1] framework and three.js [19] renderer, with all of the computations being performed on the client side. The time required to load a model in Chrome Browser on a computer with AMD FX(tm)-8320 Eight-Core Processor, Nvidia GeForce GT 710 and 16GB RAM can be seen in Table I. It is dependent on the resolution of the model, and it is important to note that over 2/3 of the loading time is spent on creation of the k-D trees.

Apart from the manual region selection, the user can precisely select the desired regions of interest by using the proposed selection modifiers. Figure 5 shows the application

Fig. 5: The average orientation filter uses the average normal of the (a) loosely selected surface and filters out all the facets whose deviation from the normal exceeds the threshold angle. The used thresholds are (b) 90 deg and (c) 30 deg.

Fig. 6: Orientation based selection selects all regions of the object which are not exceeding the threshold angle to the reference vector. The used threshold angles are (a) 90 deg, (b) 30 deg and (c) 10 deg.

Fig. 7: Application of the edge filter fill convert the (a) manual edge selection into an edge selection which has consistent distance from the selected edge. Displayed are results for (b) 30 and (c) 3 neighbors.

Fig. 8: Axis aligned bounding box selection using bounding box (a) $[0, 50, 0, 50, 0, 50]$, (b) $[0, 100, 40, 60, 0, 100]$ and (c) $[25, 75, 25, 75, 25, 75]$.

Fig. 9: Using the full coverage of the set it is possible to reveal the small areas which have not been covered by the chosen optimal viewpoint set, visible in (a). In combination with the coverage overlay ((b)-(h)), it is possible to check which viewpoints could be adapted in order to include the missed area, or where a new viewpoint should be added.

of the average orientation filter with various thresholds. It can be noted that, unlike the oriented plane filling, the filter will only discard the over-selected facets, and not add new ones. Orientation based selection will select all of the facets whose orientation is within the threshold angle to the reference orientation vector, however it is not sensitive to the plane within which the facet lies, as can be seen in Figure 6. Edge filtering will identify the facets lying within selected region, filter out any facets which are outside of the selected distance, and add facets lying within the neighboring distance from the edge. Once the edge is filtered, the filter can be re-applied with varying distance in order to adjust the selection. It can be noted that for low distances, the filter will only select facets whose centroids are closest to the edge. Therefore, it may ignore facets which touch the edge with only one vertex

because their centroid might be further than those with two vertices on the edge, as can be seen in the Figure 7. Finally, the axis aligned bounding box selection makes it easy to select particular regions of an object with a free form surface such as tanglecube, visible in Figure 8. The bounding box values depend on the size of the model, i.e. model units. In this case, the tanglecube model has the size of $100 \times 100 \times 100$ units.

Using the viewpoints and their corresponding coverage gives the user an insight into the impact of every viewpoint and the amount of the coverage overlap between the points. It becomes of particular use when evaluating the quality of the generated optimal plan and deciding whether a viewpoint should be replaced by another one. While the next best view algorithm used by [11] will surely converge, it is often the case that the latter viewpoints contribute a low number of newly covered facets. Therefore a stopping threshold may be introduced. Figure 9 shows a spring model, for which the stopping threshold was set to 99% of the surface. The optimal viewpoint set contains 9 viewpoints, 7 of which are visible from the shown angle. It can be seen that the set does not cover a small part of the third and fourth ridge, due to the stopping threshold.

VI. CONCLUSION AND FUTURE WORK

The presented tool for surface inspection planning replicated the three main parts of the physical inspection planning process; inspection object exploration, inspection region selection on the 3D model, and viewpoint selection. The inspection object exploration provides a quick and easy way for the user to evaluate the requirements posed by the object geometry in order to inspect the object completely. The inspection region selection is made easier by offering a variety of filters to make the selection faster as well as more precise. While the general object exploration and inspection region selection could have been partly done using tools such as [4], [8] or [3], the presented work combined useful features in one place, and exposed them keeping the needs and intuition of an inspection planning engineer (e.g. precise bounding box selection so that it could be done based precise specification, or selection of a specific edge, not all). Furthermore, straightforward viewpoint creation and modification, as it has been presented here has not been available in any of the above mentioned tools. Finally, the tool is not restrained by the geometrical complexity of the object and makes it easier for the engineer to create an inspection plan for complex objects.

The user can choose to use the tool in combination with inspection planning algorithms or as a standalone tool. In the first case, the user can evaluate the quality of the set of viewpoint between which the planning algorithm would choose, as well as the quality of the optimal inspection result. The quality evaluation method of the set is left to the user, however, using the tool, it can be based on the spatial distribution of the viewpoints, orientation relative to the surface, coverage, and coverage overlap between viewpoints (imaging redundancy). If needed, any of the viewpoints can be discarded, or the new ones can be added to the viewpoint set. In the second case, no

viewpoint candidates or optimal viewpoints will be available, but the user can create a completely new set and evaluate its coverage.

The presented work is just a tip of the iceberg and can be extended further in order to make itself useful in all inspection planning aspects. It has been discussed with experts and the presented functionalities have been rated as useful, however, a more extensive user study should be conducted in order to define the strengths and weaknesses more precisely from user point of view. The most prominent feature to be implemented in future is illumination placement. Current inspection planning algorithms do not offer automatic illumination placement. However, since the optimization is based on the coverage information generated by ray tracing, there is no reason why the coverage computation could not take the influence of a fixed illumination into account. Finally, by further use of the possibilities provided by the web-based implementation, the presented inspection planning tool has the potential to evolve into a collaborative platform allowing multiple users to work on the same object.

REFERENCES

- [1] AngularJS, 2019. Available at <https://angularjs.org/>.
- [2] Babtec, 2019. Available at <https://www.babtec.de/en/solutions/in-production-inspection>.
- [3] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In V. Scarano, R. D. Chiara, and U. Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008.
- [4] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, 2019.
- [5] C. K. Cowan and P. D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):407–416, 1988.
- [6] M. Gronle and W. Osten. View and sensor planning for multi-sensor surface inspection. *Surface Topography: Metrology and Properties*, 4(2), 2016.
- [7] S. Irgefnried, I. Tchouchenkov, and H. Wörn. Cadavision: A simulation framework for machine vision prototyping. In *Brno University of Technology, Faculty of Information Technology: CSSim 2011, 2nd International Conference on Computer Modelling and Simulation. CD-ROM : Brno, Czech Republic, 5 - 7 September 2011*. Brno University of Technology, Faculty of Information Technology, 2011, pages 59–67.
- [8] S. Irgefnried, H. Wörn, S. Bergmann, M. Mohammadikajii, J. Beyerer, and C. Dachsbacher. A versatile hardware and software toolset for computer aided inspection planning of machine vision applications. In L. Borzowski, J. Świątek, and Z. Wilimowska, editors, *Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology – ISAT 2017*, pages 326–335, Cham, 2018. Springer International Publishing.
- [9] X. Li, W. Yu, X. Lin, and S. Iyengar. An optimization algorithm for the np-hard 3d gallery guarding problem and its application on autonomous pipeline inspection. *IEEE Transaction on Robotics*, July 2011.
- [10] M. Mohammadikaji, S. Bergmann, S. Irgefnried, J. Beyerer, C. Dachsbacher, and H. Wörn. Inspection planning for optimized coverage of geometrically complex surfaces. In *2018 Workshop on Metrology for Industry 4.0 and IoT*, pages 52–67. IEEE, 2018.
- [11] D. Mosbach, P. Gospodnetic, M. Rauhut, B. Hammand, and H. Hagen. Feature driven viewpoint placement for model based surface inspection. Unpublished, 2019.
- [12] B. T. Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, June 1975.
- [13] Plato, 2019. Available at <https://w3.plato.de/plato-inspection-plan-1137.html>.
- [14] P. Ram and K. Sinha. Revisiting kd-tree for nearest neighbor search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19*, pages 1378–1388, New York, NY, USA, 2019. ACM.

- [15] J. Reiner. Rendering for machine vision prototyping. volume 7100, 2008.
- [16] SAP Quality Management System, 2019. Available at <https://www.sap.com/products/quality-management/product-capabilities.html>.
- [17] W. R. Scott, G. Roth, and J.-F. Rivest. View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys*, 35(1):64–96, 2003.
- [18] K. A. Tarabanis, R. Y. Tsai, and P. K. Allen. The mvp sensor planning system for robotic vision tasks. *IEEE Transactions on Robotics and Automation*, 11(1):72–85, Feb 1995.
- [19] three.js, 2019. Available at <https://threejs.org>.
- [20] Ubilabs, 2019. Available at <https://github.com/ubilabs/kd-tree-javascript>.