J Real-Time Image Proc manuscript No.
(will be inserted by the editor)

Simon Kriegel · Christian Rink · Tim Bodenmüller · Michael Suppa

Efficient Next-Best-Scan Planning for Autonomous 3D Surface Reconstruction of Unknown Objects

Received: date / Revised: date

Abstract This work focuses on autonomous surface reconstruction of small-scale objects with a robot and a 3D sensor. The aim is a high quality surface model allowing for robotic applications such as grasping and manipulation. Our approach comprises the generation of Next-Best-Scan (NBS) candidates and selection criteria, error minimization between scan patches and termination criteria. NBS candidates are iteratively determined by a boundary detection and surface trend estimation of the acquired model. In order to account for both a fast and high quality model acquisition, that candidate is selected as NBS, which maximizes a utility-function that integrates an exploration and a mesh-quality component. The modeling and scan planning methods are evaluated on an industrial robot with a high precision laser striper system. While performing the new laser scan, data is integrated on-the-fly into both, a triangle mesh and a probabilistic voxel space. The efficiency of the system in fast acquisition of high quality 3D surface models is proven with different cultural heritage, household and industrial objects.

Keywords 3D modeling · Next-Best-View planning · Active vision · Laser scanning

1 Introduction

Acquisition of 3D models is essential in several different applications, such as cultural heritage digitization, rapid prototyping and reverse engineering. Beyond these classics, the demand for high quality 3D models in robotic applications such as object recognition, grasping and manipulation is growing. Today, 3D models of unknown objects are generated by hand-guided scanner systems, manipulators, for which scans are manually planned [22], or automatic modeling systems. The latter only work for very small, mostly convex objects or require a very large, fixed and expensive setup [42, 14]. Moreover, hand-guided scanning is a very tedious and time consuming task for the human and the model quality strongly depends on the skill of the operator. A robotic system, which autonomously generates 3D models of unknown objects with an adjustable coverage or quality, would be highly beneficial. Object recognition for example still performs well even if the models are not nearly complete [18]. For grasp planning, models with higher quality and coverage are required [32]. However, autonomous 3D modeling of complex objects with a robotic system requires a coupling of 3D modeling methods with autonomous view planning and collision-free path planning.

In this work, we present an approach for autonomous 3D modeling of unknown objects in real-time. The presented approach is not limited to a class of objects, like for example convex shapes. We tackle the problem of arbitrary objects by simultaneous exploration of the unknown environment and surface modeling of the desired object. The gathered information is used to iteratively find suitable scan paths based on the object shape and plan collision free robot paths for these trajectories, until the desired model quality is reached. This work is mainly focused on the active scan planning aspect, however, the used concepts for modeling and path planning are summarized and important aspects are exposed in order to give a good overview and show the interaction between the system components. The approach is evaluated on various household, industrial and cultural heritage objects using an industrial robot with attached laser-striper.

This paper is organized as follows. In the next section, we discuss related work in the field of view planning, mapping, exploration and autonomous 3D modeling. Then, in Sect. 3, an overview of our autonomous modeling system is given. The generation of models and calculation of features is described in Sect. 4, followed by our suggested approach for planning of scans

S. Kriegel · C. Rink · T. Bodenmüller · M. Suppa
Robotics and Mechatronics Center,
German Aerospace Center (DLR),
82234 Wessling, Germany
E-mail: simon.kriegel@dlr.de

in Sect. 5. In Sect. 6, the minimization of robot pose errors and in Sect. 7, the process control are described. Experimental results are given in Sect. 8, followed by a conclusion in Sect. 9.

2 Related Work

According to Chen et al. [7] active vision perception in robotics reached a peak in 1998 and became very active again in the last years due to the variety of applications such as purposive sensing, object and site modeling, robot localization and mapping, navigation, path planning, exploration, surveillance, tracking, search, recognition, inspection, robotic manipulation, assembly and disassembly. Chen et al. state that we are currently in another peak. Further, the authors point out that inspection (15 %) and object modeling (9 %) are the most addressed vision tasks.

2.1 NBV Planning for Object Modeling

The planning of sensor views based on 3D sensor data is usually referred to as Next-Best-View (NBV) planning [34]. The NBV problem has been addressed by several researchers since the 1980s but still remains an open problem. NBV means that the robot needs to decide where to position the sensor next in order to fulfill the given task. Scott et al. [34] give a good overview of model-based and non-model-based NBV algorithms (volumetric and surface-based) for object modeling. In contrast to model-based approaches, where the views can be planned offline, for non-model-based algorithms, a NBV needs to be selected in runtime since no a priori information about the target object is given. Since a six degrees of freedom (DOF) workspace allows for infinite viewpoint candidates, a search space is usually defined containing a fixed number of viewpoints. Often, this search space is represented by sampled viewpoints over a cylinder [29] or sphere model [2,40]. The viewpoints are limited by a fixed stand-off distance such that the sensor frustum encloses the whole object. Therefore, the candidate views always point to the center of the cylinder or sphere reducing the problem from six to two DOF. These approaches lack to take into account the important fact that sensor performance depends on the stand-off distance to the surface. Besides, these approaches can not be applied to objects of which the size exceeds the camera field of view (FOV).

2.2 Mapping and Exploration

In mobile robotics NBV algorithms are used for exploration of unknown environments. Low et al. [25] obtain 3D models of larger environments using a mobile platform with a 360 degree scanner, which limits the viewpoint space to a 3DOF problem. The authors introduce a hierarchical NBV algorithm by grouping neighboring views into view volumes and neighboring surface points into surfaces patches.

In order to apply information gain (IG) driven exploration, usually metric grid maps are used. There are many mapping methods that integrate range data into a voxelmap [38]. In the work of Suppa [36] different update strategies for beam based mapping relying on depth measurements are compared. The necessity to consider sensor uncertainty is demonstrated and a probabilistic approach which interprets the sensor data is utilized. Well-known 2D-mapping techniques [38] are introduced to 3D and a detailed survey of update strategies and their application in the context of robot work cell exploration is given. Suppa proposes to use a Bayesian update rule for voxels. This choice is supported by Potthast et al. [30], who prove it to be more efficient than using a simple approach for exploration of uncluttered scenes with a humanoid robot and an eye-in-hand camera. Suppa also gives a first approach how to integrate NBV planning for exploration and 3D modeling. A very similar mapping approach to 3D mapping has been proposed by Wurm et al. [45] and advanced by Hornung et al. [10], who also give a more detailed overview of the literature.

2.3 Surface Quality Measure

The goal of most volumetric NBV algorithms is to increase the knowledge about the unseen portions of the viewing volume, which is also how Banta et al. [2] define the term Next-Best-View. Even if the volumetric space is completely known, it is not ensured that the surface model, which is the desired output of 3D modeling, has reached a certain quality. However, it is very difficult to give a measure for the quality of the reconstructed object for unknown objects since no ground truth is given. So far, only few approaches [26,1,41] also consider a quality measure while planning the NBV. Massios et al. [26] were the first that used a quality criterion in addition to a visibility criterion to improve the quality of the surface. They suggest to use the angle of incidence (the angle between surface normal and viewing direction) for each voxel as surface quality. Vasquez-Gomez et al. [41] also use this quality criterion but additionally take into account the traveling distance and overlap with previous range images. They sample 80 candidate views over a sphere, which represents a sphere search space, starting at a low ray tracing solution and then evaluating the best views with a higher resolution.

All these authors lack to prove that their suggested quality criterion leads to a better reconstructed model quality of the surface model. Reaching a high quality in

the volumetric model, does not implicate a high quality for the required surface model.

Johnson et al. [12] and Mehdi-Souzani et al. [28], both investigate the measurement error for different angles of incidence. Both come to the conclusion that the noise is almost constant up to a certain angle and increases enormously at an incidence angle greater than 60°. Therefore the angle of incidence should be used as quality criterion but only as a binary. Furthermore, in [37] it is shown that the surface to sensor distance plays a very important role for laser scanners and should also be considered as quality criterion. Similarly, the evaluation of Scheibe [33] shows that the 3D point quality depends a lot more on the distance than on the angle of incidence. If the search space is restricted to a sphere like in the work above, then distance cannot be considered since it is already predefined by the sphere.

2.4 Autonomous Object Modeling Systems

Several NBV algorithms are only tested in simulation and ignore robotic aspects. In contrast, Callieri et al. [6] and Larsson et al. [21], both use an industrial robot in combination with a turntable to model the objects. The former focus on the 3D modeling but do not consider path planning aspects at all. For the latter, the user needs to manually input object size and stand-off distance for each object individually, which does not render the system autonomous. Also no processing times are mentioned. Torabi et al. [39] try to scan a set of points on the occlusion surface which they call target points. However, this is similar to other methods and they do not consider improvement of the known surface. Furthermore, the viewpoint search space is still discretized by four spheres with different directions. Between two scans, the robot waiting time is several minutes, which is far too much. Karaszewski et al. [13] introduce a measurement system to model large cultural heritage objects, where the human needs to initialize the size of the object. Karaszewski suggests that a system for 3D modeling should not depend on a robot type. In a first step areas in the boundary area and in a second step areas with low point density, are selected as viewpoint candidates. All viewpoints are simply processed without reasonable NBV selection and also no abort criterion is introduced. 3D modeling of a few cultural heritage objects is shown but the quality of them is not evaluated. The system does not seem to be optimized concerning time. For a small object, the digitization time was over 19 hours.

For high quality 3D modeling, laser sensors, which require to be moved, are used. Previous autonomous modeling systems [24,39] plan a NBV and then perform a scan by simply rotating the last joint of the robot. This is not an optimal sensor movement since it does not consider the contour of the object or the estimation of the unknown environment.

2.5 Distinction

In our work we present a complete and autonomous robotic system for real-time 3D modeling of unknown objects which aborts after a defined mesh quality is reached. The objects can be arbitrary, except for their expansion, which is limited by a bounding box depending on the robot workspace. The scan path candidates are not sampled over a sphere or cylinder model, but are directly estimated based on the shape of the partial triangle mesh, which is generated in a real-time stream during the laser depth measurement. Simultaneously, the laser range data is streamed to the probabilistic space update, which is also performed in run time. Next-Best-Scan (NBS) planning, as introduced in [19], describes the planning of continuous sensor paths such as linear robot movements. Here, the trajectory is not a fixed movement. Thus, NBS planning can be seen as an extension of the NBV problem that also requires additional collision free path planning along the trajectory. Furthermore, NBS planning allows for the usage of line range sensors, such as laser stripe profilers. However, the method for the scan path estimation can also be applied to aerial 3D sensors by using the midpoint of a scan path as NBV, which however does not ensure that the complete object is scanned. It has already successfully been applied for single viewpoint planning in object reconstruction [9] and object recognition [18].

Due to the real-time model update in this work, NBS planning can be directly performed after each scan. The time for the NBS planning is optimized by only using newly acquired data for each iteration. Our approach is a mixture of surface-based and volumetric planning and the advantages of both models are exploited. The voxel space is used for exploration and collision free path planning. The triangle mesh is used for the scan path estimation, termination criteria and saving surface features to the voxel space. During NBS selection, both aspects are considered. To our knowledge, we are the first to consider the quality of the surface model when planning the NBV or in our case NBS. Also the surface quality is used as termination criteria and therefore the desired mesh quality can be configured depending on the application for which the 3D model will be used.

The main contributions in comparison to a previous approach [19] are a real-time space update, pose error minimization, NBS selection considering both IG and surface quality and a termination criteria depending on the model completeness and point density.

3 Overview

The principal idea of autonomous modeling of unknown objects is to incrementally build a complete surface model by sweeping over the object surface with a 3D sensor. For every sweep, the already existing 3D model

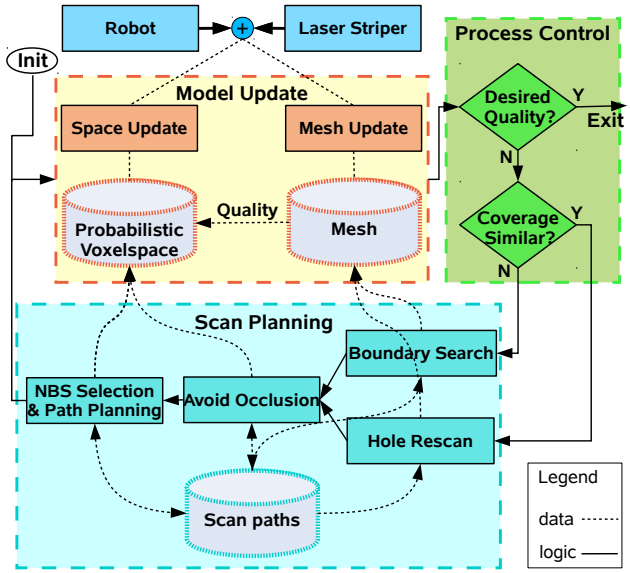


Fig. 1 Overview of autonomous 3D modeling process: 3D modeling is performed in a real-time stream during the laser scanning, scan planning is carried out after each scan.

is extended and new scan paths that potentially can further complete the model are calculated. However, calculated scan paths can cause collisions with the object, especially at more complex geometries. Therefore an additional probabilistic volumetric model is build and used during path planning to avoid trajectories through occupied or unexplored parts of the workspace. Further, model-based scan path calculation can fail, for example if there are sharp edges or multiple, separate objects. Here, exploration of unknown areas is used to iterate further.

The resulting autonomous modeling system is divided in three functional blocks: *Model Update*, *Scan Planning* and *Process Control*. Fig. 1 gives an overview of the interactions between these modules.

The *Model Update* (yellow box) incrementally integrates the new range measurements. In this work two different 3D models are used, a triangular mesh and a probabilistic voxel space (PVS). Here, the mesh represents the application goal and is used to find boundaries, to calculate a sampling quality and for surface trend estimation. The PVS represents the exploration aspect and is used for occlusion avoidance, collision free path planning and storage of the surface quality of measured depth points.

In the *Scan Planning* module (blue box), NBS candidates are calculated from either the mesh boundaries or detected holes. Then, occlusions are avoided and the PVS is used to rate the candidates and to select the collision free NBS.

Finally, the *Process Control* (green box) monitors the modeling process, switches the planning mode and terminates the process, if the model has a certain completeness and quality level. The system is initialized by setting part of the PVS to the state *unknown*. The volume of that part has known size and position and covers at least the goal object.

As prerequisite, the modeling system is connected to a 6DOF industrial robot with attached laser striper. Further, it is assumed that the robot-sensor-system provides a real-time stream of globally aligned range data, i.e. for each range measurement a transformation is given that allows for the calculation of 3D points in a global coordinate systems. For the assumed eye-in-hand sensor configuration, the global transformation is the robot pose at measurement time with an additional calibration transformation. In real systems however, the robot pose has a limited precision that can result in a misalignment of the range data. Thus, the pose error is minimized by registration of the range data.

4 Model Update

In this work, two models are generated from the measurement data, a triangle mesh and a PVS. Both models are used in every planning step for the calculation and selection of scan paths. Further, the PVS is needed for collision-free path planning. Therefore, new measurements have to be integrated into the existing model at every iteration.

For the incremental generation and refinement of a triangle mesh, the *streaming surface reconstruction* approach is used. This algorithm has originally been presented by Bodenmüller [5] for instant model generation and visualization with hand-held scanner systems. For iterative generation and update of a PVS an octree-based voxel space with Bayes-update is used. This approach was already presented by Suppa [36] for work cell exploration with industrial robots.

In the following, the used mesh generation and the PVS update are summarized and important aspects are outlined. Finally, the extraction of surface quality and the mapping to the PVS is explained.

4.1 Mesh Update

In this work, a triangle mesh is defined by a set of vertices $\mathbf{v}_i$ with corresponding surface normals $\mathbf{n}_i$ and a set of directed edges e_j, which represent the line segments connecting two adjacent vertices $\mathbf{v}_a$ and $\mathbf{v}_b$ along the surface. Each edge e_j has a direction $\mathbf{e}_j = \mathrm{dir}(\mathbf{v}_a, \mathbf{v}_b)$ with

$$\mathrm{dir}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{b} - \mathbf{a}}{|\mathbf{b} - \mathbf{a}|} \tag{1}$$

The triangle faces are not stored explicitly but for each edge e_j two additional vertices $\mathbf{v}_l$ and $\mathbf{v}_r$ (see Fig. 2), which close the adjacent triangle to the left and to the right with respect to the direction, are assigned.

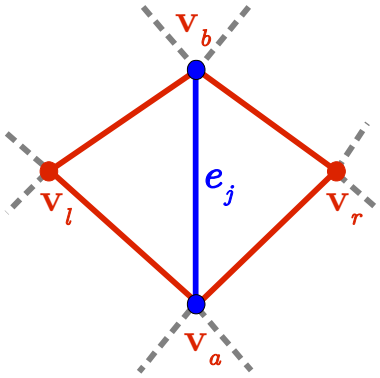


Fig. 2 An edge of the triangle mesh consists of two vertices defining the line segment (blue) and two additional vertices closing the adjacent triangles (red) to the left and right.

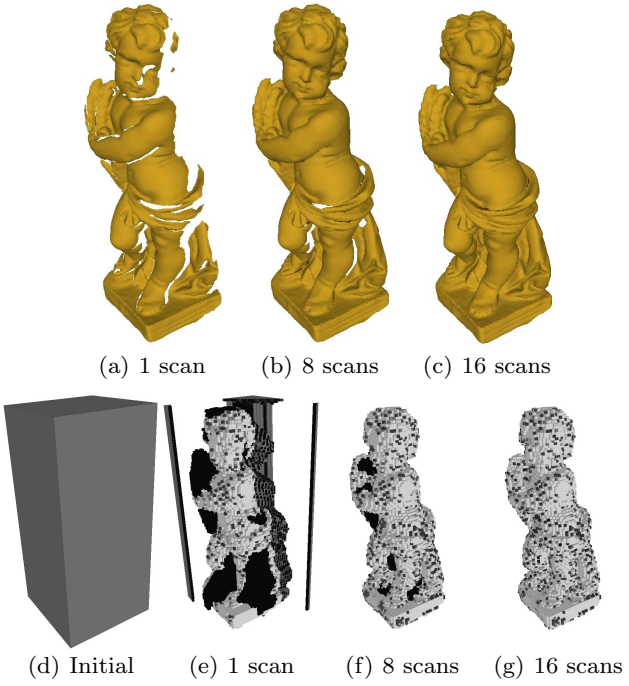


Fig. 3 Model Updates. Upper row: The mesh is updated during each laser scan. Lower row: The PVS is initialized with unknown voxels and then updated during each scan. The probabilities are color coded from black (almost free), through gray (unknown) to white (occupied). Free space is transparent. The updates are shown after 1, 8 and 16 scans respectively.

The reconstruction consists of three principal stages, the *density limitation*, the *normal estimation* and the *mesh generation* step, as already presented in [5]. Each range image is converted into a set of 3D points and incrementally inserted into the model. At insertion of a new point, it is tested if the point is not closer than a distance R_r to any model point and rejected if the test fails. The test can be performed by requiring an empty ball neighborhood with radius R_r. The ball neighborhood is the subset of points that are within a bounding sphere centered at the regarded point and with radius R_r . This *density limitation* limits the overall Euclidean point density of the model. In the *normal estimation* step, the ball neighborhood with radius R_n is calculated for each newly inserted point. The surface normal is estimated using principal component analysis with a weighted covariance matrix for all vertices within the neighborhood. If the surface normal is a robust estimate, the point is forwarded to the *mesh generation* step. During the mesh generation stage, the new points are inserted as vertices of the emerging mesh. For every newly inserted vertex a *localized triangulation* is performed by projecting a local ball neighborhood with radius R_m to the tangent plane of the new vertex and a re-triangulation of this 2D subset. Finally, triangles are recalculated from the changed edges. The result is a mesh with an edge length between R_r and R_m. Fig. 3 shows the example mesh updates after 1, 8 and 16 scans performed for a putto statue.

The reconstruction approach was originally designed for out-of-stream[1] data processing from arbitrary manual scanner systems. Hence, the approach is not restricted to a certain type of 3D sensor, but only requires that the sensor data can be transformed into a point set with additional line-of-sight for each point. Since out-of-stream processing requires fast computations, the whole approach is based on the usage of localized data. All calculations use only a local subset of the data, namely the ball neighborhood, which has an upper bound in size, due to the point density limitation. The only global operation is the query of the neighborhood for each inserted point, which is accelerated by an octree data structure. The octree is used, because it is the best trade-off concerning computational effort between insertion of new data and neighborhood query. The point set is stored in the containing leave voxel of the octree. A query operation is performed by finding all voxels that intersect with the neighborhood sphere and testing all points in the voxels. Since the initial *density limitation* results in an upper bound for the number of points per voxel, the complexity of query depends only on the octree search of the voxel. However, the latter increases only logarithmic for unbounded volumes and is constant for a bounded volume.

[1] out-of-stream processing denotes the processing of data directly from a real-time data stream, e.g. the live stream of a 3D sensor or camera

4.2 Probabilistic Voxel Space Update

A voxel space is the partitioning of $\mathbb{R}^3$ space into discrete elements, the so-called voxels. The size of the elements is denoted as resolution of the space. In a PVS, each voxel holds a state value, representing the likeli-

hood that the cell is free or there is an obstacle, usually starting with a complete unknown space.

Here, the voxel space is built incrementally by updating it with each measurement. Therefore, the measurement beam for each pixel of a range image is calculated. Having very similar tasks to accomplish as Suppa [36], we follow his choice to use Bayes-update. For mapping, forward sensor models are calculated from inverse models in a preprocessing step and stored in a hash table. Thus, each measurement beam induces a state of occupancy and freedom for the hit cells. These induced states are combined with the cell's current states and stored as their new states. When using Bayes-update, the states represent a transformation of the likelihood quotient and can be interpreted as a measure for the cell's probability to be occupied. As the Bayesian update requires statistically independent measurements, not every sensor beam is used. Therefore, all view positions and directions of each measured beam are saved to a list and similar measurements during the update are rejected [36]. This is reasonable, since due to usually high resolution of 3D sensors, neighboring rays often intersect the same voxels. This has clearly the effect of speeding up the update significantly, compared to a naive update that uses all sensor beams.

The PVS is implemented using a dynamic octree, similar to the one used for the mesh update. The octree provides fast operations for both insertion and query of data at a low memory consumption. However, the data structure of the PVS is kept independent of the one used for the mesh generation, since it contains different data and the resolutions of the two are adapted to different requirements.

Fig. 3 shows an example of an initially unknown space and the progress after 1, 8 and 16 scan sweeps with a laser striper. The probabilities are color coded from black (almost free), through gray (unknown) to white (occupied). Free space is transparent.

Note that the resolution of the voxel space has various effects on the algorithm and has to be chosen with caution. The smaller the resolution is, the more accurate the modeled object or environment will be within the voxel space. Disadvantages of such a small resolution are the consumption of more memory, the increasing computation time and less information per voxel (concerning quality, see Sect. 4.3). Contrary, a larger resolution results in less localized information per voxel, causing a large occupied area around the object. Thus, scan paths have to be further away from the object.

4.3 Surface Feature Update

The desired output of the autonomous 3D modeling is a complete, high-quality 3D triangle mesh. Therefore, local features describing the completeness and quality are derived from the mesh. In detail, two features describing the quality are used here: a local sampling density and an incidence angle between a measurement beam and the surface model. Concerning the completeness, the percentage of border edges is calculated. The features are calculated for each voxel of the PVS and stored additionally to the probability of occupancy, since the NBS selection processing step (see Sect. 5) is based on the PVS. The combination of the proposed features enables for the calculation of an optimality-criterion with respect to an expected improvement of the surface quality of already scanned areas, as outlined in the next section.

4.3.1 Sampling Density

Let N_{act} be the number of points within the *normal estimation* neighborhood with radius R_n and N_{max} the maximum possible number of neighbors according to the reduction radius R_r of the *density limitation*. Then we call the quotient

$$d(\mathbf{v}_i) = \frac{N_{act}}{N_{max}} \qquad d \in [0,1] \tag{2}$$

the local point density. It describes the sampling density around a vertex $\mathbf{v}_i$ and can therefore be used to measure sampling sufficiency. The close-packing of spheres theorem yields (cf. [5])

$$N_{max} = \frac{\sqrt{2}\pi \cdot R_n^2}{R_r^2}. \tag{3}$$

Let in the following i denote (the index of) a voxel. Then, the average density $\bar{d}_i$ within voxel i is calculated by averaging over each d of the vertices within that voxel.

4.3.2 Average Surface Normal

Also, we later use an angle of incidence for filtering voxels in the surface quality determination. Since the exact calculation of the incidence angle of a surface beam with the triangle mesh proved to be too time consuming, we calculate and store an average surface normal $\bar{\mathbf{n}}_i$ for each voxel i. It is extracted directly from the mesh by averaging over all vertex normals within voxel i. If the space resolution is set properly, $\bar{\mathbf{n}}_i$ can be used for incidence angle estimation of a sensor beam with the surface. Note that a similar approach has been proposed by Vasquez-Gomez [41], which lacks the more precise information of a mesh, but simply estimates an average normal from neighboring voxels.

4.3.3 Amount of Border Edges

In order to account for the completeness, the percentage of mesh border edges b_i is calculated for each voxel.

A border edge is an edge to which a triangle is only assigned on one side and on the other side there is nothing (see Fig. 2). For a complete mesh of an object, no border edges should exist. Therefore the border edge percentage is an indication whether this voxel requires rescanning. It is defined by the number of border edges within a voxel divided by the total number of edges:

$$b_i = \frac{N^i_{border}}{N^i_{total}} \qquad b_i \in [0,1]. \tag{4}$$

5 Scan Planning

Based on the triangle mesh and PVS from the current scan, further scans need to be planned for a robot-in-the-loop. First, a *Boundary Search* is performed resulting in scan path candidates. Second, a NBS is selected and a collision free path is planned for a laser striper. Additionally, a hole rescan is performed after the surface model is fairly complete.

5.1 Boundary Search

In this section, the *Boundary Search*, which is introduced by Kriegel et al. [17], is described in more detail. It is similar to the approaches which find the NBV by an occlusion edge [27,29], with the difference, that not only occlusion edges are detected but also the known object shape is applied to the estimated surface trend. A boundary is defined as a set of connected edges that satisfy two requirements: the edges all have a similar orientation and each edge lacks an assigned triangle, either on the left or on the right side. The *Boundary Search* only considers newly acquired triangle mesh data.

The *Boundary Search* consists of two stages, which are described in detail in the following. During the first stage, the *Boundary Detection*, the different boundaries of the object are detected. Then, the *Surface Trend Estimation* searches for a boundary region of vertices for each boundary in order to fit these to a quadratic patch. Finally, scan paths can be determined (see Sect. 5.2) using the surface trend with the constraint that the sensor looks perpendicular to the estimated surface and there is an overlap with the previous scan.

5.1.1 Boundary Detection

Depending on the edge orientation, four boundary types *left*, *right*, *top* and *bottom* are defined in the following. Therefore, the given triangle mesh, assumed to be part of a complete object, is transformed into the coordinate system of the sensor. Then, the side on which the boundary is located, will be referred to as boundary type. Here, the sensor coordinate system is defined so that the z-coordinate represents the viewing direction, the x-coordinate is to the left and the y-coordinate is in the up direction. Thus the left and right boundaries are approximately in direction of the y-axis and top and bottom in direction of the x-axis. The difference of the two boundary pairs is the side on which no mesh exists. Some example edges in partial meshes of a putto statue and a pneumatic filter are depicted in Fig. 4.

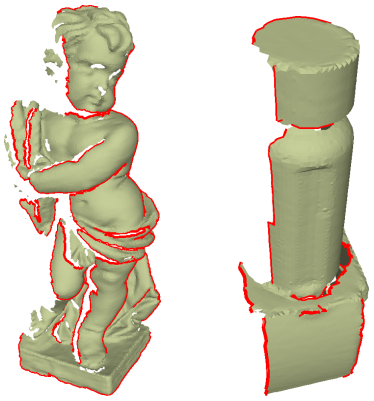


Fig. 4 Boundaries (thick red lines) detected in partial meshes of putto statue (left) and pneumatic filter (right).

Algorithm 1 Recursive boundary detection

```
B : initially empty set of boundary edges
e : an edge
p : penalty for boundary end detection (initialized with 0)

BOOL findBoundaryRecursive(e)
if isBorderEdge(e) then
  Add e to B
  for all edges ẽ connected with e do
    if isBorderEdge(ẽ) and not ẽ ∈ B then
      /* Compares angle according to Equation (5) */
      if α > α_t then
        p = p + 1
      else
        p = 0
      end if
      if p > p_t then
        return (size(B) ≥ b_min)
      end if
      return findBoundaryRecursive(ẽ)
    end if
  end for
end if
return false
```

Algorithm 2 Border edge detection

```
BOOL isBorderEdge(e)
/* see Fig. 2 */
return e has v_l xor e has v_r
```

For each newly acquired edge e we try to find a continuous boundary by locally walking over the mesh border. The approach is described in Alg. 1 exemplary for left boundaries. On success, the function returns a set of edges $\mathcal{B} = \{e_1, \ldots, e_m\}$ describing the boundary. However, only boundaries with at least b_{min} edges are considered. An edge e is a border edge, if there is only one triangle assigned, as described in Alg. 2. Otherwise,

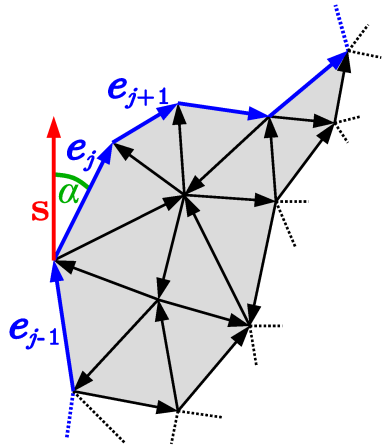


Fig. 5 The detection of a left boundary in the mesh is shown: for each edge along the boundary the angle α between the sensor axis $\mathbf{s}$ (red, in this case y-axis) and the vector of the current edge e_j is computed and compared with threshold α_t.

e is added to the boundary list $\mathcal{B}$ and all incoming and outgoing edges $\tilde{e}$ connected with e are inspected. These are all edges e_{j-1} or e_{j+1} as in Fig. 5, which provide a common vertex with e, either $\mathbf{v}_a$ or $\mathbf{v}_b$. Among these, for all border edges the angle α between a sensor coordinate axis $\mathbf{s}$ (for left boundary: y-axis) and the directed vector $\tilde{\mathbf{e}}$ of the current edge candidate $\tilde{e}$ (see Fig. 5) is determined as follows:

$$\alpha = \arccos\left(\frac{\mathbf{s}\cdot\tilde{\mathbf{e}}}{|\mathbf{s}||\tilde{\mathbf{e}}|}\right). \tag{5}$$

The angle α is utilized to detect the boundary type and abort if the end of the boundary type is reached. The sensor axis $\mathbf{s}$ is defined by the y-axis for left and right boundaries and the x-axis for top and bottom. Fig. 5 shows a possible left boundary represented by blue vectors. The angle α is determined for the current edge e_j. This will be continued for the outgoing boundary edge e_{j+1} and incoming boundary edge e_{j-1}. If the angle α is larger than a threshold α_t then a penalty value p, which is initialized with zero, is increased for the currently observed boundary. The penalty allows for slight deviations of the orientation of the sensor axis, as can be seen in Fig. 5. Thereafter, the angle will be calculated accordingly for the edge $\tilde{e}$ (previously e_{j+1}), which is next in the edge chain. The penalty is reset to zero once an edge with a good angle is found. The algorithm aborts the boundary detection once the penalty exceeds a threshold p_t and then the procedure is repeated in the other direction with the previous edge e_{j-1} of the edge chain.

Finally, a boundary is considered as left boundary if it comprises a certain number of edges b_{min}. If the number of edges is too low, then a reasonable surface trend estimation is difficult. The procedure for determining the right, top and bottom boundaries is performed accordingly during the same iteration. Furthermore, the direction of the normals along the boundary are compared with the sensor viewing direction and discarded if they are from the opposing side.

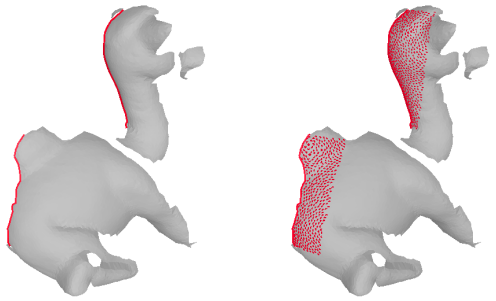
5.1.2 Surface Trend Estimation

After the boundaries are detected, boundary regions need to be found, in order to be able to estimate the surface trend. A boundary region consists of several vertices in the area of the boundary. The surface trend or also trend surface describes the general shape of a surface. It is often applied for fitting and interpolating regression surfaces to a smoothed representation of area data. It is based on the assumption that a spatial arrangement of a surface can be represented by a defined geometric function (for details see [44]). The surface trend can be applied for prediction of the unknown surface of an object or environment. Thus, it will be used in our work to estimate the object shape. The surface trend is estimated for each boundary individually, since complex objects cover several different geometrical shapes and cannot be approximated by one surface trend. Chen et al. [8] also uses the surface trend for reconstruction of unknown objects. However the surface trend is simply estimated for the complete object. This method mostly works for simple objects, e.g. cylindrical objects, but has problems with more complex shapes.

First, a boundary region needs to be found which can be used to estimate the surface trend. Thus, for each boundary a region growing, which is limited by a bounding box, is performed. The region growing starts with the center vertex of the boundary and iteratively adds all neighbor vertices which are within the bounding box. The bounding box, e.g. for the left boundary, is defined by a bottom and top margin, the y-values of the first and last vertex of the boundary edge chain and a right margin (x-value), a fraction of the total horizontal expansion of the mesh. Fig. 6 shows an example of two left boundaries, which are detected for a partial mesh of a camel statue, and the corresponding boundary regions. The region growing is performed inward to the known part of the mesh. It is important that the boundary region is not selected too large, otherwise the surface trend does not estimate the curvature well, since a more general shape is estimated. Second, the surface trend of the unknown area beside the boundary is estimated using the boundary region. We choose a simple approach to fit all the vertices $\mathbf{v}_i = [x_{\mathbf{v}_i}, y_{\mathbf{v}_i}, z_{\mathbf{v}_i}]^t$ of the boundary region, to a quadratic patch:

$$z = f(x_{\mathbf{v}_i}, y_{\mathbf{v}_i}) = a_1 x_{\mathbf{v}_i}^2 + a_2 x_{\mathbf{v}_i} y_{\mathbf{v}_i} + a_3 y_{\mathbf{v}_i}^2 + a_4 x_{\mathbf{v}_i} + a_5 y_{\mathbf{v}_i} + a_6. \tag{6}$$

A quadric is chosen since it is of low order and it gives a good estimate to whether a boundary mesh area, which is not subject to too much change, is curved outward or inward in the direction of the unknown area. Therefore, the approximate curve (quadratic patch) in the



(a) Left boundaries (b) Boundary regions

Fig. 6 Example of two left boundaries obtained from a partial camel mesh. A region growing is performed for both regions in order to fit a quadratic patch.

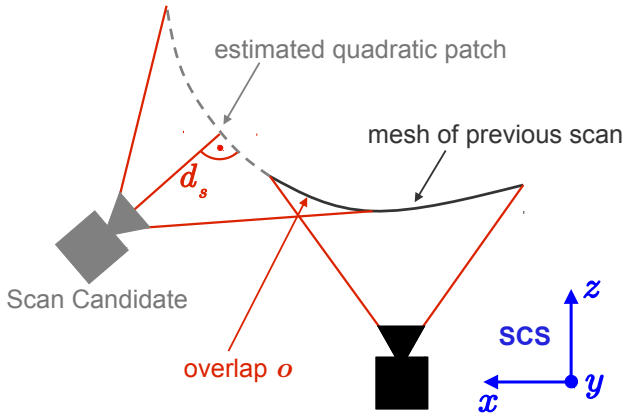


Fig. 7 Scan Path Calculation: the FOV overlaps with the mesh from previous scans by factor o. The scan path looks perpendicular onto the estimated quadratic patch at the optimal sensor distance d_s. The axes of the SCS only refer to the initial scanner pose (black sensor head).

unknown area can be estimated quickly, which suffices to determine viewpoints according to the trend of the surface.

5.2 Scan Candidate Calculation

The surface trend estimation enables the calculation of continuous scan path candidates. A scan path is defined by a start and an end position and a fixed orientation. The number of views can later be adjusted by the speed the robot moves along the shortest path between start and end point. It is usually selected slow enough to ensure a high point density for qualitative mesh generation. Fig. 7 shows a possible scan path from the top, which views the center of the estimated surface, which is beside the scanned region, at a perpendicular angle. The sensor coordinate system (SCS) is also shown which is for the initial scanner pose (bottom right). Furthermore, the required overlap o with previous scan data and the distance d_s are adjustable. d_s is selected according to the sensor depth of field, in order to obtain optimal measurements, in contrast to a sphere search space, which is fixed based on the sphere center. For laser stripers, which usually have a very narrow depth of field, the distance is very important. If the sensor is too close or too far away, nothing can be measured. Furthermore, an overlap with the previously scanned mesh is required in order to obtain a complete model and for registration of the different scans. The overlap o represents the percentage of the part of the FOV which views the partial mesh of previous scan data. First a viewpoint is determined, which could be used for aerial 3D sensors. Second, based on the viewpoint a scan path is determined, which consists of several viewpoints. The length of the scan path depends on the expansion of the partial triangle mesh of the object (for explanation see below).

5.2.1 Viewpoint calculation

Starting at the detected boundary, candidates for a viewpoint are determined by calculating points and normals along the estimated quadratic patch. For simplification, we start at a point along the boundary, which is the midpoint of the first and last vertex of the boundary. Possible surface points $\mathbf{s}_i = [x_{\mathbf{s}_i}, y_{\mathbf{s}_i}, z_{\mathbf{s}_i}]^t$ are calculated in the direction of the unknown area by inserting $x_{\mathbf{s}_i}$ and $y_{\mathbf{s}_i}$ into Equation (6). When performing this for the left boundary we keep $y_{\mathbf{s}_i}$ constant and increase $x_{\mathbf{s}_i}$ by a step size. Then the surface normal $\mathbf{n}_i = [x_{\mathbf{n}_i}, y_{\mathbf{n}_i}, z_{\mathbf{n}_i}]^t$ is calculated from the derivatives of Equation (6):

$$\mathbf{n}_i = \begin{pmatrix} \frac{\partial f}{\partial x_{\mathbf{s}_i}} \\ \frac{\partial f}{\partial y_{\mathbf{s}_i}} \\ -1 \end{pmatrix} = \begin{pmatrix} 2a_1 x_{\mathbf{s}_i} + a_2 y_{\mathbf{s}_i} + a_4 \\ a_2 x_{\mathbf{s}_i} + 2a_3 y_{\mathbf{s}_i} + a_5 \\ -1 \end{pmatrix}. \quad (7)$$

The $z_{\mathbf{n}_i}$ is set to -1 since the viewing direction of the scanner is described by the positive z-axis and surface points are in the opposite direction. For this surface point a candidate viewpoint is calculated at the optimal sensor distance d_s from the curve and in direction of the normal. The candidate viewpoint is required to have an overlap of o with the previous mesh, with the constraint, that the angle between two consecutive viewpoints does not exceed a limit. If the candidate viewpoint does not comply with the constraints, then for the left boundary the value of the step size is decreased and a new candidate is determined or the algorithm aborts. For the left boundary, $x_{\mathbf{s}_i}$ is increased and a new surface point is calculated until the desired overlap o is reached. Finally, the mesh and the scan candidates are transformed back into the world coordinate system. When using a sensor, which measures 3D range data without moving the sensor, these viewpoints can be utilized directly and one could proceed with the NBS selection of Sect. 5.4.

5.2.2 Scan path calculation

If we apply a laser scanner, which only measures a stripe with depth values, then not only a single viewpoint but a real scan path is required. Therefore, we calculate a continuous scan path along the boundary using the fixed orientation of the calculated viewpoint. The fixed orientation of the laser striper is defined by a SCS as in Fig. 7. The z-axis is in the viewing direction. The y-axis is along the boundary but not necessarily parallel. Here, a scan path is represented by a linear movement (shortest path between two points) of which the length is adapted to the current known object surface. The inverse of the quadric patch surface normal $\mathbf{n}_i$ (see Equation (7)) represents the viewing direction (z-axis). For the y-axis of the SCS, we use the boundary direction $\mathbf{d}_b = \mathrm{dir}(\mathbf{v}_1, \mathbf{v}_m)$, which is the normalized vector connecting the first $\mathbf{v}_1$ and last vertex $\mathbf{v}_m$ of a boundary. As the boundary direction is independent of the estimated quadratic patch normal, $\mathbf{d}_b$ is not necessarily perpendicular to $\mathbf{n}_i$. Therefore, the x-axis is defined as vector product between the two and then the y-axis by:

$$\underbrace{(\mathbf{d}_b \times -\mathbf{n}_i)}_{x\text{-axis}} \times \underbrace{-\mathbf{n}_i}_{z\text{-axis}} . \tag{8}$$

The scan path is in direction of $\mathbf{d}_b$ at distance d_s. In order to also scan as much as possible of the rest of the mesh, the length of the scan path is chosen larger than just the boundary direction. Therefore, we define a plane, having the boundary direction $\mathbf{d}_b$ as normal, and intersecting the midpoint of the boundary. Now, the length of the scan path is defined by the minimum and maximum distance of all mesh vertices to the plane. This proved to be more efficient, since by scanning along the complete mesh and not just the boundary, other unknown parts of the object were also scanned. Otherwise more scans, which require time for planning and moving to, were required.

The benefits of using the *Boundary Search* for NBS selection in comparison to a sphere search space are, that overlap is already considered and therefore NBVs or NBSs will be beside the known region. This leads to a short robot movement from the current NBS to a subsequent NBS. The major advantage though is that the scan path candidates are not predefined but estimated from the current sensor information and therefore are adapted to the actual object shape. The search space is not restricted, which allows for better modeling results, since the distance and grazing angle of the sensor to object are not restricted and regions which cannot be seen from a sphere can also be viewed.

5.3 Hole Rescan

When the surface model is fairly complete, typically some small holes may remain in the model due to occlusions or objects with difficult surface properties such as blackness or reflectivity. In that situation, the scan path candidate calculation according to previous sections is not optimal, since usually the calculated paths view larger regions than necessary and might not be able to view a complete hole with optimal viewing angle. Therefore, when the coverage of two subsequent scans is similar (see Sect. 7), the remaining scan paths from the boundary search are discarded and for each hole an adequate linear scan path is calculated.

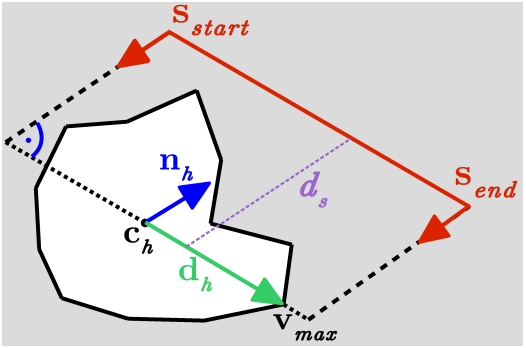


Fig. 8 For each hole (white area) within the measured surface (gray), a scan path is determined in direction of the largest expansion of the hole $\mathbf{d}_h$ in inverse hole normal $\mathbf{n}_h$ direction and at optimal sensor distance d_s.

Holes are detected by iterating over all edges of the triangle mesh and finding a closed loop of border edges. For each border edge, neighboring border edges are successively searched, which together form a path of edges. This path is considered to be a hole if the path is closed. Then, for each hole, the center $\mathbf{c}_h$ and normal $\mathbf{n}_h$ is determined, as suggested by Loriot et al. [24], by averaging over all vertices and normals of the hole boundary. A scan path is determined along the largest hole direction $\mathbf{d}_h = \mathrm{dir}(\mathbf{c}_h, \mathbf{v}_{max})$, which we define by the direction between $\mathbf{c}_h$ and the boundary vertex $\mathbf{v}_{max}$, which is furthest away from the center (see Fig. 8). Finally, start and end point of the scan path are calculated by adding a relative threshold of 10 % on each side and multiplying the normal with the sensor distance d_s:

$$\mathbf{s}_{start} = \mathbf{c}_h - 1.1 \cdot \mathbf{d}_h + d_s \cdot \mathbf{n}_h \tag{9}$$

$$\mathbf{s}_{end} = \mathbf{c}_h + 1.1 \cdot \mathbf{d}_h + d_s \cdot \mathbf{n}_h. \tag{10}$$

The scan direction is the inverse of the hole normal $\mathbf{n}_h$. Additionally, for holes with similar center position and orientation, a combined scan path is determined. Of course, one could close the holes in a postprocessing step. However, this would distort the real object contour and is not acceptable for accurate 3D modeling. After the mesh is fairly complete, we only perform the hole detection once and scan holes until the desired coverage is reached. Thereby, real holes are only scanned once.

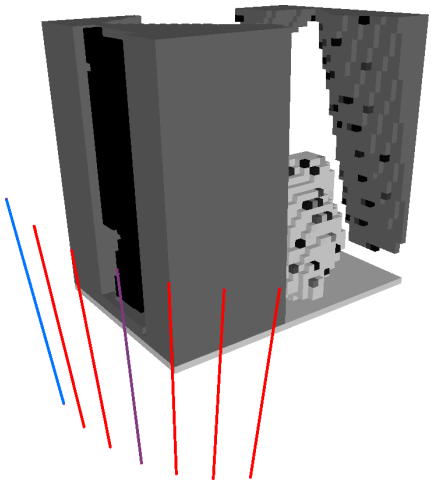


Fig. 9 A scan path candidate (purple, center) in collision is re-planned by rotating the paths around the object part of interest. The purple and all red scan paths are in collision with the platform or not reachable by the robot workspace. The blue scan path represents a occlusion and collision free path viewing the same area at the cost of a worse angle. Here, the PVS is partly explored for a camel.

5.4 Next-Best-Scan Selection

Based on the list of scan path candidates, which are either generated during the *Boundary Search* or the *Hole Rescan*, a NBS needs to be selected.

In order to cope with objects with complex geometry, which are not mostly convex and contain several self-occlusions, first the scan path candidates are re-planned avoiding occlusions and collisions based on the PVS. Thereby, similar to Prieto et al. [31], all scan paths, which are occluded by other parts, are iteratively rotated around the part of the object, which is supposed to be scanned, until it is not occluded anymore. Fig. 9 shows an example during autonomous modeling of a camel. The purple scan path, which is in collision, is re-planned resulting in the blue path which is occlusion and collision free. All other paths in the figure are in collision with the platform on which the object is positioned or not reachable by the robot workspace. Note that the scan paths in Fig. 9 represent the center of the sensor system, of which the dimensions need to be considered during collision free path planning. If the incidence angle is too high for reasonable scan quality, this scan path candidate is removed from the storage.

Second, a NBS is selected based on a novel utility function, which considers both surface quality and IG. In order to get a measure of the IG of a single viewpoint candidate, usually ray tracing in the voxel space is performed. Some NBV methods simply count the number of the viewed unknown voxels [4,43]. Thereby sensor uncertainty is not considered and only the first intersected unknown voxel of the beam is observed. In [19], the entropy reduction is added up for each intersected voxel and the scan path with highest expected IG is selected as NBS. This works very good for the first scans. However, once the voxel space is sufficiently explored but the required quality is not yet reached, this measure is not applicable since most voxels are almost free or almost occupied. Therefore, as measure to select a NBS, we use the entropy of the volumetric model e_v but add a surface quality value q_s to a utility function $\mathrm{f}_{\mathrm{utility}}$, which is determined for each scan path candidate. The weighting between the two can be adjusted depending on the task:

$$\mathrm{f}_{\mathrm{utility}} = \underbrace{(1-\omega)\cdot e_v}_{\text{Exploration}} + \underbrace{\omega\cdot(1-q_s)}_{\text{3D Modeling}}. \tag{11}$$

The function consists of an exploration and 3D modeling part. The exploration part selects a NBS which views the sum of voxels with the highest expected IG. The 3D modeling part chooses a NBS which views previously scanned voxels with poor mesh quality. For the first scans, the exploration part needs to be weighted higher in order to get a rough model of the unknown object. Once, a rough triangle mesh is obtained, the 3D modeling part needs to be considered more, since now the mesh quality should be addressed. Therefore the weight ω is selected such that it depends on the scan number n_s:

$$w(n_s) = \frac{\frac{n_s}{n_q}}{\frac{n_s}{n_q}+1}. \tag{12}$$

For n_q a value of five is selected, which means that after five scans, the exploration and the 3D modeling part are weighted equally. For all further scans, 3D modeling is considered more. Without the weight ω, the algorithm needed a lot of scans to get a rough model of the object all around, which is not very efficient.

For each scan path candidate, rays are cast onto the PVS based on the sensor model of the applied sensor. An important information that can be derived from a PVS is the IG. Information or entropy of a sensor view is the sum of weighted probabilities of all voxels in that view. The total entropy e_v for a candidate view is defined by the mean entropy over all voxels i, which are intersected by a beam until an occupied voxel is reached:

$$e_v = -\frac{1}{k}\sum_{i=1}^{k} \underbrace{p_i \log(p_i)}_{\text{occupied}} + \underbrace{(1-p_i)\log(1-p_i)}_{\text{free}}, \tag{13}$$

where k is the total number of intersected voxels.

The probability p_i represents the probability of voxel i to be occupied. If a voxel is free ($p_i = 0$) or occupied ($p_i = 1$) then the entropy reduction is zero.

Additionally to the entropy, for each voxel i, which is intersected and contains surface features (see Sect. 4.3), a surface quality q_i is determined:

$$q_i = \begin{cases} \lambda\cdot b_i + (1-\lambda)\cdot \bar{d}_i & \text{if } \theta < 70° \\ 0 & \text{otherwise} \end{cases}. \tag{14}$$

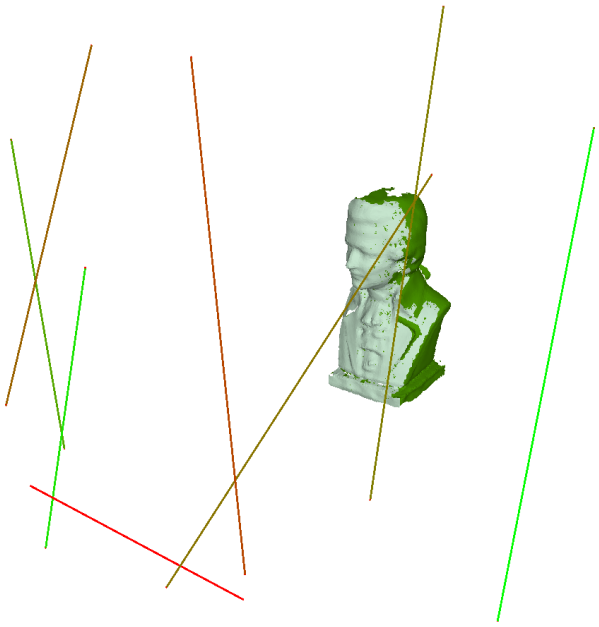


Fig. 10 A Mozart bust is initially scanned from the front (light green mesh). After calculating scan path candidates based on the *Boundary Search*, the scan with the highest rating is selected as NBS. Here, the scan candidates are color coded from low (red) to high (green) utility rating. Based on the NBS, in this case the rightmost scan path, the mesh is extended (dark green) and the NBS planning continues until the required model quality is reached.

The incidence angle θ is calculated by forming the dot product between the simulated beam and the average voxel surface normal $\bar{\mathbf{n}}_i$. If $\theta \geq 70°$ then $q_i = 0$, since we assume that a re-scan of this surface area will not increase the quality of the surface model due to the large angle of incidence (see Sect. 2.3). If θ is below 70°, the surface quality q_i is determined by weighting the border edge percentage b_i and the average relative point density $\bar{d}_i$. For λ a value of 0.7 proved to be good, which means that completeness is weighted higher than point density. The quality of the complete surface model q_s is calculated by getting the average of q_i:

$$q_s = \frac{1}{k} \sum_{i=1}^{k} q_i. \tag{15}$$

After determining a rating for each scan path candidate according to Equation (11), the scan path with the highest value represents the NBS. Fig. 10 gives an example for the NBS selection during the autonomous modeling of a Mozart bust. The utility rating of each scan path is represented by color coding from low (red) to high (green). In this case, the rightmost scan candidate is selected as NBS and the process will continue until the desired model quality is reached.

5.5 Path Planning

In order to be able to safely move the robot to and along the NBS, collision free path planning is performed based on rapidly-exploring random trees [20] and probabilistic roadmaps [15]. The paths are planned using the PVS, which describes the unknown area, and a robot model containing CAD data of the sensor setup and constant models of the environment. This allows for avoiding any type of collision.

Some scan paths may not be reachable by the robot or the space may not be free yet. Since our approach does not restrict the search space and thus scan paths are generated which can be very close to the object, collision avoidance and robot reachability are very relevant. Therefore, based on the PVS, which is updated during each laser scan, we plan a collision free path to the start position of the NBS and along the complete continuous scan path. It is mandatory that the PVS update (see Sect. 4.2) frees as much unknown space as possible to allow for collision free path planning for as many scan paths as possible. If the NBS is not reachable by the robot, it is either discarded, if it goes through an obstacle, or marked as currently in collision and kept for later iterations, if it intersects with unknown space. Then the scan path from the stack with the second highest rating is selected as NBS. However, if at least 80 % of the NBS is reachable, then a NBS for this part is performed.

6 Pose Error Minimization

The absolute positioning errors of most robots are far too high for a precise 3D modeling. Especially orientation errors easily lead to a misalignment of the range images in global space. This results in a noisy or corrupted mesh at the model update. The pose error can be minimized by using the range image data, since the 3D sensor usually has much higher accuracy.

In this work, a variant of the well-known Iterative Closest Point (ICP) algorithm [3] is used to correct the robot pose of each new scan, also denoted as 3D registration. For aerial 3D sensors the ICP can be used directly on each range image. With line sensors, however, this is not possible. Therefore, in this work, the data of a complete scan path is merged, a local mesh is generated and registered to the global mesh with ICP. In every iteration of the ICP, correspondences between the local and global mesh are searched by assigning all points in a certain radius as corresponding points. Here, we use a radius of three millimeters. From the correspondences a least-square estimation of the transformation is calculated. The estimated transformation is applied to the range images and finally the corrected range images are integrated into the global model (as described in Sect. 4.1).

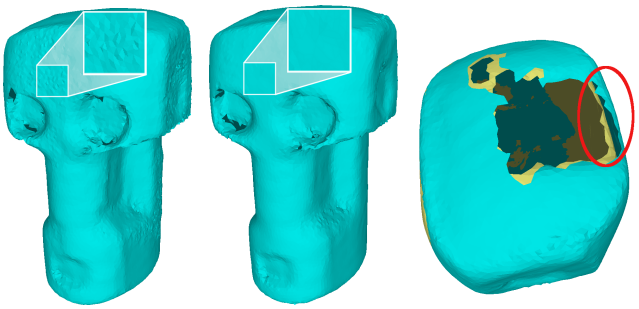


Fig. 11 The difference in modeling of a sensor head. From left to right: 3D model without ICP, with ICP (two iterations) and comparison between two iterations (blue) and converged ICP (yellow). Note the considerable gap between the surfaces marked with the red ellipse.

The ICP requires a sufficiently structured surface and an overlap of at least 50 % between the new data and the global model. The latter can be considered during the calculation of NBS, as described in Sect. 5.1. The surface structure, however, depends of course on the object. Especially technical objects often contain poorly structured or symmetric parts that cause the ICP to an overfitting of the local mesh. Since it can be assumed that the pose error is small, the ICP can be aborted after two or three iterations, minimizing possible overfitting.

Fig. 11 shows the difference between modeling without ICP (left) and its application with two iterations (center). The application of ICP leads to a smooth surface, as shown in the zoomed area. However, as can be seen in Fig. 11 right, the overall model tends to be contracted if too many iterations of the ICP are carried out. For the sensor head of Fig. 11 the distortion was approximately four millimeter.

The proposed ICP-based pose correction helps to improve the automated meshing, since it reduces the influence of pose errors on the model generation. However, it is based on the assumption that the relative pose error during a single scan is low. This holds for most industrial robots with serial kinematics, since here a local small movement is more accurate than a global movement, which requires large movements in the base axes. For other systems, such as mobile platforms, the assumption does not hold and thus the ICP correction is either limited to aerial 3D sensors or additional pose sensing is required. Here, a possible extension is to estimate local movement with a feature tracking [35] and register this data via ICP [11]. Nevertheless, the model-based correction always can result in a slight overfitting and thus distort the final 3D model. Hence, for an optimized model it is proposed, that all scan data is optimized in a post-processing with a bundle adjustment.

7 Process Control

The control terminates the process, if the desired quality is reached, and switches the scan planning mode, when the quality of subsequent scans is similar.

It is difficult to find a reasonable termination criterion if the object is unknown and no ground truth is given. Torabi et al. [39] point out that most previous NBV methods lack a termination criterion, which considers the actual object shape coverage. They abort if a maximum number of views is reached [40], if the model does not change significantly anymore after a scan [43] or if all air points [21] or boundaries [17] have been scanned once. Torabi also suggests that the model is complete if no boundaries in the point cloud remain. However, even if the object is complete within the point cloud or voxel space, the surface model can still contain several holes. A triangle for the mesh cannot be generated if no neighborhood point can be found within a certain radius. In [19], the percentage of border edges in the mesh is used as a factor to estimate the mesh completeness. However, this measure does not give a good estimate on the actual completeness percentage of a partial mesh, as here the area, which is not filled, is relevant. The size of the holes cannot be estimated based on the border edges, since a certain number of border edges, can describe several holes with very small area or also one hole with a very large area.

In this work, we determine the surface area A_{filled} of all triangles in the mesh and estimate the surface area for each hole individually, which is summed up to the total area A_{empty} of all holes. The mesh coverage is estimated by:

$$\hat{c}_m = \frac{A_{filled}}{A_{filled} + A_{empty}} \qquad \hat{c}_m \in [0,1] \tag{16}$$

The filled mesh area is the sum of the area of all n triangles, which is half the cross product of the two spanning vectors of a triangle:

$$A_{filled} = \frac{1}{2} \cdot \sum_{i=1}^{n} \|\mathrm{dir}(\mathbf{v}_a, \mathbf{v}_b) \times \mathrm{dir}(\mathbf{v}_a, \mathbf{v}_c)\| \tag{17}$$

In order to determine A_{empty}, the surface area A_{hole} for each hole area is approximated. Thereby, the hole center $\mathbf{c}_h$ is determined by averaging over all m vertices along the edge of the hole

$$\mathbf{c}_h = \frac{1}{m} \sum_{i=1}^{m} \mathbf{v}_i \tag{18}$$

and the hole area A_{hole} is estimated by forming a triangle for each edge with the hole center $\mathbf{c}_h$. The sum of the hole area A_{hole} of all k holes describes the estimated empty mesh area

$$A_{empty} = \frac{1}{2} \sum_{i=1}^{k} \|\mathrm{dir}(\mathbf{c}_h, \mathbf{v}_{i-1}) \times \mathrm{dir}(\mathbf{c}_h, \mathbf{v}_i)\| \tag{19}$$

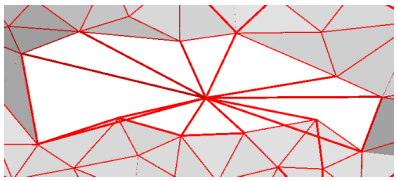


Fig. 12 The area of an example hole is estimated by forming triangles to the hole center for each edge along the hole. The method does not describe the actual hole area accurately but gives a quick and good estimate on it.

Fig. 12 shows how for each edge along an example hole triangles are formed by creating edges towards the center. Certainly we could also fill the holes with a standard bicubic method [23] and determine the area of the filled hole. However, bicubic hole filling is complex and computational expensive. Since we estimate the mesh coverage $\hat{c}_m$ after each scan, the suggested method seems sufficient concerning time and result.

With a high neighborhood radius during the mesh generation, a mesh with 100 % completeness can easily be achieved, at the cost of losing the details. Therefore, simply evaluating the mesh coverage such as in [16, 39, 19] is not always reasonable. As our mesh generation inserts new vertices even in areas, where the mesh is complete, a combination of measuring mesh coverage and point density is important. In this work, the algorithm aborts if a certain mesh coverage $\hat{c}_m$ and average relative point density $\bar{d}_{mesh}$ over the complete mesh is reached. If both are never reached due to object geometry and sensor restrictions, then the algorithm aborts after a predefined number of scans. For the objects in our experiments, 30 scans seemed to be a good number.

Further, the process control switches from the scan path candidate generation based on *Boundary Search* to *Hole Rescan*, when the estimated coverage $\hat{c}_m$ stagnates. Therefore the mesh coverage for the previous scan $\hat{c}_m^{i-1}$ and the current scan $\hat{c}_m^i$ are compared. If the estimated coverage increases less than 1 %, i.e.

$$\hat{c}_m^i - \hat{c}_m^{i-1} < 0.01 \tag{20}$$

then *Hole Rescan* is performed.

8 Experiments and Evaluation

In this section, our new method is compared with the preceding NBS approach [19] by applying it to nine different objects from three different fields of application. The comparison is with respect to the number of scans, total execution time, average point density, object coverage and modeling error. For the experiments, an industrial laser striper, the ScanControl 2700-100 from Micro-Epsilon, attached to the flange of a Kuka KR16 industrial robot, is used (see Fig. 13). The robot allows for exploration of objects of a maximum size of about 300 mm×300 mm×500 mm, which are placed on a fixed platform. However, most but not all scan paths during the experiments could be reached. In the following, first the hardware and experimental setup is explained. Second, the performance of the system is discussed, using exemplary results on nine different objects.

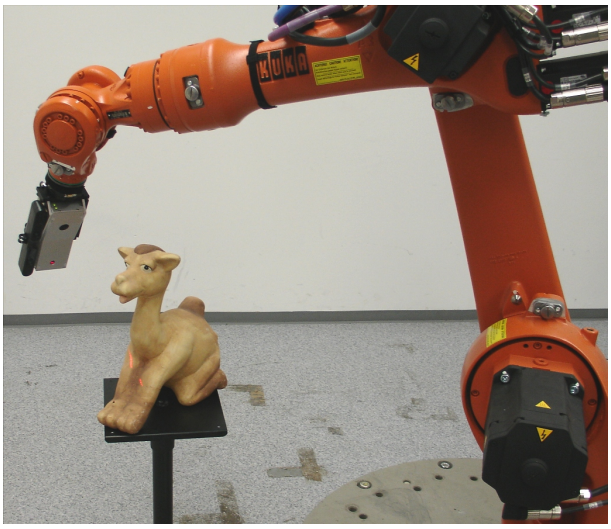


Fig. 13 Overview of experimental setup: laser striper is attached to an industrial robot in order to autonomously generate a 3D model of an unknown object placed on a platform.

8.1 System Setup

The KR16, with an absolute positioning error of 1 mm, is used to move the sensor in between and during scans and to get the sensor pose during the scan. The pose of the robot is sent from the robot controller (KRC4) to an external computer at 250 Hz, where it is synchronized with the laser stripe profiler, which measures 640 depth points at a frequency of 50 Hz. The PC used for processing has Quad Xeon W3520 2.67 GHz CPUs and 6 GB RAM. The working area of the laser striper is very limited with a depth of field of 300 mm to 600 mm and a relatively narrow FOV angle of 14.25°, which requires accurate view planning. Therefore, a sensor distance d_s of 450 mm, which is the center of the depth of field, is selected for the scan path calculation (see Sect. 5.2). The maximum measuring error is approximately 0.5 mm for the laser striper and 2.5 mm for the complete system. The complete system error was determined by obtaining a depth image from both sides of a very thin board with triangle patterns and calculating the distance between the patterns in the range data. Due to the sensor base distance, some object parts such as cavities simply cannot be scanned. Today, good postprocessing techniques are available for filling holes considering the object shape. These will not be used here, since then the scan data is manipulated. Often autonomous 3D modeling systems are evaluated on objects with complex shapes but reasonable surface properties such as

Fig. 14 The objects used during experiments. *Top*: camel, Mozart and Zeus (cultural heritage). *Middle*: cookies, dog spray and Santa Claus (household). *Bottom*: control valve, filter and pressure valve (industrial). The largest object is the camel with 340 mm height, the smallest the control valve of 95 mm.

for cultural heritage objects. However, especially industrial parts pose a challenge concerning surface properties due to dark and reflective parts. Therefore, the presented autonomous 3D modeling system will be tested on different industrial, household and cultural heritage objects (see Fig. 14). Due to the different surface properties of the objects areas, for each area an individual termination criteria is defined.

8.2 Parameter Settings

For all experiments, the mesh generation is configured with a limitation radius R_r=1 mm, a normal neighborhood radius R_n=3 mm and a mesh neighborhood radius R_m=3 mm, resulting in an edge length in between 1 mm and 3 mm. The parameters are bound to the accuracy and resolution of the scanner system. Hence, a lower resolution is not feasible due to the described robot and sensor accuracy. A full overview of the mesh generation parameters and an extensive analysis on parameter variation and coupling is presented in [5].

A too small PVS resolution increases the computation time and decreases the number of mesh vertices that are used for per-voxel feature calculation. A PVS resolution of at least R_m guarantees that the upper bound for the per-voxel point density, controlled by R_r is sufficient. Here, a PVS resolution of 10 mm is chosen for the current system, which is a suitable trade-off between performance, detail and robustness.

For the *Boundary Search*, the angle threshold $\alpha_t = 45°$, which allows for a variation of the boundary and no mismatch with different boundary types. The parameters p_t and b_{min} need to be adjusted depending on the object size, the resolution of the triangle mesh and the desired number of boundaries. If these values are selected to be too high, then only very few boundaries are detected and vice versa. This is relevant since many boundaries result in many scan paths. These require a longer computation time if each scan is simulated in order to select the best one. We used values of $p_t = 5$ and $b_{min} = 12$. For our nine test objects, for which the maximum size is limited by the hardware, the selected parameters performed well and were robust. However, for very large objects, the parameters would need to be adjusted, which has not been examined in this work.

Table 1 Evaluation parameters used for comparison

Variable	Description
n_s	number of scans
t	total execution time in min
$\bar{d}_i$	average relative point density over all vertices
$\hat{c}_m$	estimated object coverage rate in %
c_b	actual completeness rate (with bottom) in %
c_a	actual completeness rate in %
$\bar{e}$	coordinate root mean square error in mm

8.3 Evaluation Criteria

The evaluation criteria used for comparison are listed in Tab. 1. The point density $\bar{d}_i$ and estimated coverage $\hat{c}_m$ are determined for the final model as suggested in Sect. 4.3 and 7. For comparison, a ground truth mesh model is obtained with an expensive hand-guided scanning system, which comprises a maximum sensor error of 0.05 mm and a system accuracy of 0.1mm. The manual ground truth model generation took approximately 40-70 min per object including scanning, repositioning, manual registration and postprocessing. Due to manual postprocessing, the ground truth models could be completed. The ground truth allows for evaluation of the actual object completeness c_a and the coordinate root mean square error $\bar{e}$, which seems feasible since the hand-guided system has an accuracy, which is higher by approximately factor 25. The actual completeness c_b (with bottom) is also measured by comparing the generated mesh with the ground truth model, where a mesh exists for the bottom area. This is not feasible for actual completeness evaluation, since the objects for the autonomous 3D modeling are placed on a platform and therefore the bottom of the objects cannot be scanned. But it seems fair to use c_b for evaluating the performance of our termination criteria, the estimated coverage $\hat{c}_m$, which also considers the hole at the bottom.

8.4 Comparison with previous method

In [19], we have already shown that when using the *Boundary Search* (see Sect. 5.1) for view planning, more

Table 2 Comparison of our autonomous 3D modeling method simply based on IG and based on a combination of IG and quality with streaming space update and ICP registration for different objects (for parameter description see Tab. 1). The results are given for both methods: IG (left), IG/Quality (right).

	Camel	Mozart	Zeus	Cookies	Spray	Santa	Control	Filter	Pressure
n_s	30/**16**	15/**14**	**14**/15	6/**5**	30/**14**	6/6	**9**/11	17/**15**	30/**14**
t	27.5/**9.3**	10.8/**5.5**	10.7/**6.5**	3.0/**1.5**	21.7/**5.8**	3.1/**1.9**	5.5/**3.8**	11.7/**6.1**	19.6/**5.4**
$\bar{d}_i$	0.51/**0.64**	0.52/**0.60**	**0.54**/0.52	0.42/0.42	0.58/**0.62**	0.37/0.37	0.50/**0.58**	0.46/0.46	0.42/**0.46**
$\hat{c}_m$	81.1/**84.4**	85.8/**88.7**	83.4/**86.4**	78.6/**86.6**	74.5/**82.5**	90.7/**91.3**	69.5/**77.2**	75.8/**81.9**	64.9/**77.2**
c_b	80.7/**84.2**	90.5/**93.0**	88.4/**90.5**	82.3/**88.6**	**90.7**/89.8	**93.0**/91.6	74.7/**78.1**	88.1/**88.6**	81.9/**83.0**
c_a	93.6/**95.2**	98.5/**99.9**	97.8/**99.2**	90.0/**99.5**	**97.7**/97.2	**99.5**/99.1	94.6/**97.9**	92.7/**93.2**	85.4/**86.2**
$\bar{e}$	**0.76**/0.80	0.80/**0.76**	0.70/**0.64**	0.81/**0.70**	**0.95**/0.98	1.10/**0.73**	**0.91**/0.96	0.79/**0.77**	1.58/**1.50**

complete 3D models can be generated in shorter time than with a standard sphere search space. Here, we compare our previously presented method *IG* which considers IG as NBS selection criterion [19] with our new method *IG/Quality* which considers both IG and quality as suggested by the utility function in Equation (11). For the NBS selection simply based on IG, ω is set to zero in the utility function. The new method *IG/Quality* also features space update in a real-time stream and pose error minimization by scan matching. For better comparison, the process control including termination criteria of the novel method is used for both. The results for both methods applied to the test objects are compared in Tab. 2 (left value: *IG*, right: *IG/Quality*).

8.4.1 Run Time

In order to acquire dense sampled surface data, the laser striper is moved along a commanded linear trajectory at a low speed. Here, the fusion between robot pose and range measurement or at least the temporal labeling and logging must be performed in real-time. All other processing steps could basically be executed between the movements, as a quasi-offline processing. This would, however, result in a bad performance for the overall system. Therefore, in this work, the updates of the triangle mesh and the PVS are performed out-of-stream during the scan movement. The other steps, namely the scan registration, the NBS planning, and the path planning are performed after the scan movement, since they require the complete set of new data from the scan. Although all these steps have to perform operations quickly on a steadily growing data set, the computation time did not increase with the mesh or PVS data size. However, the time for the NBS selection increased with a larger number of scan path candidates.

As described in the previous sections, modeling and planning are tackled in a *soft real-time* way. Instead of analyzing the complete data set, only local areas with bounded data size are concerned for modeling and current changes are regarded for planning. The few necessary global operations on the data set are accelerated by an octree data structure.

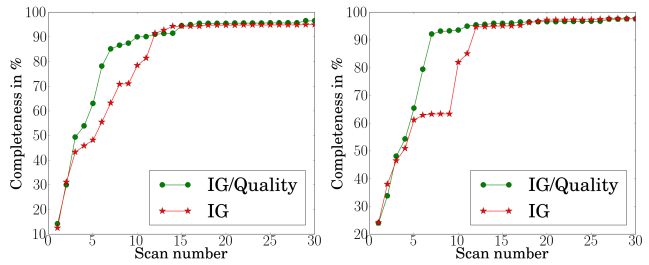
Table 3 Average processing time for each module per iteration in seconds and percentage of total iteration time

Moving robot in between scans	6 s	35.3 %
Streaming modeling and scanning	7 s	41.2 %
Scan registration	1 s	5.9 %
NBS planning	2 s	11.8 %
Path planning	1 s	5.9 %
Complete Iteration	17 s	

In Tab. 3, the iteration times for the individual modules are listed. During our experiments one complete iteration of the *IG/Quality* method took 17 s on average. The time for moving the robot according to the path planner to the start position of the NBS trajectory took 6 s. The modeling is performed during an average of 7 s while moving the robot and acquiring an average of 224000 depth points per scan. The scan registration, NBS planning and path planning took only 4s, which represents only 23.5 % of the total iteration time. Hence, the robot had to wait 4 s after each scan, 60 times less than for the autonomous 3D modeling system of Torabi et al. [39], which does not consider real-time constraints. Their system requires 4 min for processing the data and NBV planning between two scans on a similar PC with Quad i5-760 2.8 GHz CPUs and 8 GB RAM. For our previous approach [19], the space update was not performed during, but after each scan. Further, independence of measurements as outlined in Sect. 4.2 were not considered, which led to a multiple of space updates. Therefore, the waiting time was 20 s. This is an average speedup of approximately 16 s per iteration to the previously presented method [19]. For the nine objects, the total execution time t for *IG/Quality* is significantly less by an average factor of 2.3. This is also due to the fact that on average less scans are required.

8.4.2 Model Quality

As can be seen in Tab. 2, for the camel, dog spray and pressure valve, the desired quality (coverage and point density) was never reached for the *IG* method and the algorithm aborted after the fixed number of 30 scans. This also shows that the *IG* approach does not really



(a) Camel (b) Filter

Fig. 15 Development of the actual mesh completeness c_a during the acquisition process of the camel and filter object

aim at increasing the quality. The relative point density $\bar{d}_i$ is only better for the previous approach for the Zeus object. The model error $\bar{e}$ is mostly below one millimeter and only less for the *IG* method for the camel, dog spray and control valve. For the pressure valve, the error is a lot higher than one millimeter, which is probably due to the many reflections during the scanning of the object. As $\bar{e}$ is always significantly lower than the complete system error of 2.5 mm, this shows that the ICP algorithm aided to reduce the model error. The estimated coverage $\hat{c}_m$ and actual completeness c_b (including bottom part) are mostly similar. $\hat{c}_m$ underestimates c_b by an average of 5.4 %. This shows that the suggested estimated coverage (see Sect. 7) is a good termination criterion.

Furthermore, the actual completeness c_a is better with the new approach for all objects except for the spray and Santa Claus. However, for the spray more than twice as many scans were performed with *IG* and still the desired point density is never reached. Fig. 15 shows the development of the completeness c_a of the triangle mesh exemplary for the camel and filter object after each scan. The suggested termination criterion was not used for better comparison, but the system simply aborts after the predefined 30 scans. Here, the *IG/Quality* method reaches a significantly higher completeness than *IG* after a few scans. For both objects, the completeness is more than 20 % higher after seven scans. The completeness starts to stagnate between 12 and 17 scans. After that, for *IG/Quality* the completeness is only between zero and two percent higher. However, the last few percent contain the object details and cost the major amount of time to address. For objects with complex geometry, it takes a human operator less time to scan the first 90 % than the last 10 % using a hand-guided scanning system.

The final 3D surface models for the new method are shown in Fig. 16. As can be seen from Tab. 2, the range of the actual mesh completeness considering bottom part c_b varies depending on the size of the bottom area of the object, which cannot be scanned. For the Santa Claus, the completeness is very high since the bottom area is small. The control valve has a large floor

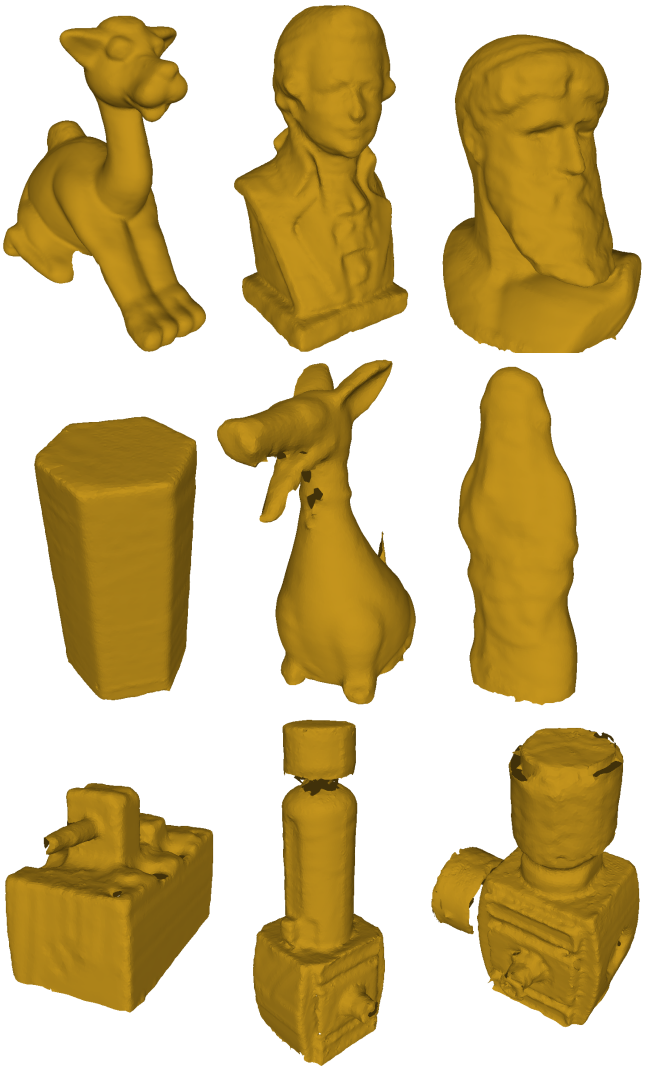


Fig. 16 3D surface models of objects of different areas: cultural heritage (*top*), household (*middle*), industrial (*bottom*)

space. The actual completeness c_a of the models generated with the new approach is over 99 % for the Mozart, Zeus, cookies and Santa Claus objects. This indicates that these objects are basically complete. However, the other objects were more difficult to scan, which is indicated by the visible holes in the final surface models (see Fig. 16), and did not reach such a high completeness. The camel contains areas at the bottom that the sensor cannot reach. Furthermore, the industrial objects and the dog spray (middle row, center) are very reflective and contain dark areas, which are difficult even for the laser to handle. For the pressure valve (bottom right in Fig. 16), the back part of the clock could not be scanned due to sensor characteristics, which is the reason for the lowest c_a of all objects. Also, the industrial objects contain actual holes, e.g. for screws. These are not considered by a completeness criterion, as it assumes that the object can be 100 % scanned.

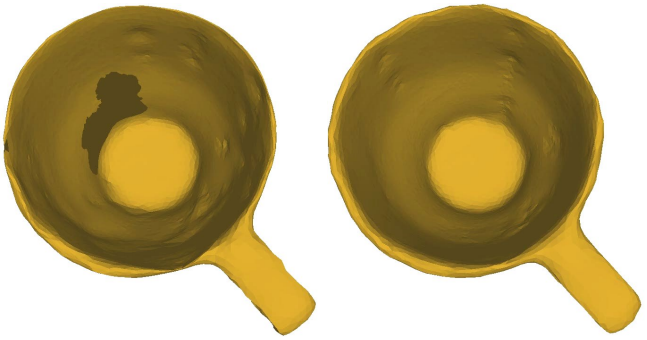


Fig. 17 3D models of coffee cup from top with standard ICP registration (*left*) and extension by considering normals (*right*).

Nevertheless, the obtained models contain most of the details of the object and proved to be good enough to be used for reliable object recognition [18], even when the termination criteria is decreased and therefore less scans are required. Also, if objects cannot be scanned perfectly due to object shape or sensor limitations, it is unlikely that other vision systems could measure the corresponding part of the object. In contrast, some remaining holes in the models might also be avoided if the values for the termination criteria are increased, which, however, would lead to more scans and a longer execution time. The models are of significantly higher quality than with RGB-D or ToF sensors. However, they are not as good as with the expensive hand-guided scanning system. But the acquisition time is a lot lower.

We also tried to acquire a 3D model of an IKEA coffee cup, which represents a more complex challenge as the side walls are very thin and the inside embodies a deep concave area. At first, the mesh generation and ICP algorithm both had difficulties with the thin walls. Therefore, the ICP algorithm was extended by comparing the surface normals of the corresponding points and discarding correspondences with an angle difference of more than 60°. This allowed for avoiding registration of scans obtained from the inside and outside of the cup. During the mesh generations stage, the same criterion was applied in order to avoid meshing of inside and outside parts. Fig. 17 shows the final 3D surface model of the coffee cup with standard ICP registration (left) and the suggested extension which considers surface normals (right). For standard ICP, the inside of the cup is fitted to the outside in the upper area. Therefore outside and inside overlap, which can be seen by the dark area (backside of outside mesh). With the extension, this problem does not occur and also the walls are not as thin in the upper area as can be seen in the left figure. Without ICP, the pose error caused a noisy mesh with double walls. A 3D model of the coffee cup was obtained with 14 scans within 4 min. Most of the scans were performed for the inside and handle of the cup, which were difficult to completely acquire due to the triangulation principle of the sensor. A completeness of 98.3 % (without bottom) was reached and most importantly the complete inside of the cup was scanned.

Overall, our results represent a good trade-off between surface quality and duration time. We have shown that with our new approach, 3D models are obtained a lot faster and also the quality (completeness rate and point density) of the surface model is higher. The complexity of the autonomously scanned objects by our system is an advancement to objects that can be automatically scanned by state-of-the-art systems.

9 Conclusions and Future Work

In this work, an autonomous 3D modeling system, consisting of an industrial robot and a laser striper, for efficient surface reconstruction of unknown objects is presented. Both 3D models, a probabilistic voxel space and a triangle mesh, are updated in real-time during the laser scan and are then iteratively used for scan planning. Thereby, only local changes in the models are considered for computational optimization. Iteratively, from a set of possible scan paths, which are estimated based on the surface shape, a NBS is selected, considering both surface quality and space information gain, and a collision free path is planned. Errors in the robot pose are minimized by applying a version of the ICP algorithm. The system terminates if a desired model quality is reached. No human interaction is required and the final 3D mesh can be used directly, e.g. for object recognition or grasping. The proposed method proved to outperform existing methods in the depicted scenarios. The new selection criteria led to a faster acquisition of a complete model and the on-the-fly data integration improved the performance compared to a former approach. Also, our autonomous system was on average approximately 11 times faster than a human using a hand-guided but more accurate scanning system. It has been shown that complete and high quality models of different cultural heritage, household and industrial objects could be acquired autonomously by the system within a few minutes. The time varied between just above one minute for the Santa Claus object, which is small and has a simple shape and almost 10 minutes for the camel, which is a lot larger and contains several cavities. More importantly, the robot waiting time between two scans was on average only four seconds, which aided the real-time 3D model acquisition.

Future work will comprise model generation of objects with a more complex geometry, texture mapping and a final bundle adjustment of the collected scan patches. Also, in order to get the complete model, e.g. the bottom areas, the objects should be repositioned and further scanned for modeling of previously unmodeled parts. Furthermore, some modules such as NBS

selection or occlusion avoidance could be further accelerated by exploiting graphics hardware.

Acknowledgements This work has partly been supported by the European Commission under contract number FP7-ICT-260026-TAPAS. The authors would like to thank the editor and all the reviewers for their constructive comments. Our special thanks go to Daniel Seth for his support with the octree structure, to Andreas Dömel for his help with the path planner, to Klaus Strobl for helping with the sensor calibration and to Zoltan-Csaba Marton for good ideas and feedback.

References

1. Albalate, M.T.L., Devy, M., Martí, J.M.S.: Perception planning for an exploration task of a 3d environment. In: IEEE ICPR, pp. 704–707. Washington, DC, USA (2002)
2. Banta, J.E., Wong, L.R., Dumont, C., Abidi, M.A.: A Next-Best-View System for Autonomous 3-D Object Reconstruction. IEEE TSMC **30**(5), 589–598 (2000)
3. Besl, P.J., McKay, N.D.: A Method for Registration of 3-D Shapes. IEEE PAMI **14**(2), 239–256 (1992)
4. Blaer, P., Allen, P.K.: Data acquisition and view planning for 3-d modeling tasks. In: IEEE/RSJ IROS, pp. 417–422. San Diego, California, USA (2007)
5. Bodenmüller, T.: Streaming Surface Reconstruction from Real Time 3D Measurements. Ph.D. thesis, Technische Universität München (TUM) (2009)
6. Callieri, M., Fasano, A., Impoco, G., Cignoni, P., Scopigno, R., Parrini, G., Biagini, G.: RoboScan: An Automatic System for Accurate and Unattended 3D Scanning. In: IEEE 3DPVT, pp. 805–812. Thessaloniki, Greece (2004)
7. Chen, S., Li, Y., Kwok, N.M.: Active vision in robotic systems: A survey of recent developments. IJRR **30**(11), 1343–1377 (2011)
8. Chen, S.Y., Li, Y.: Vision sensor planning for 3-D model acquisition. IEEE TSMC **35**(5), 894–904 (2005)
9. Foix, S., Kriegel, S., Fuchs, S., Alenyà, G., Torras, C.: Information-gain view planning for free-form object reconstruction with a 3d tof camera. In: ACIVS, *LNCS*, vol. 7517, pp. 36–47. Springer, Brno, Czech Republic (2012)
10. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots **34**(3), 189–206 (2013)
11. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., Fitzgibbon, A.: Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In: ACM UIST, pp. 559–568. New York, NY, USA (2011)
12. Johnson, A.E., Hoffman, R., Osborn, J., Hebert, M.: A System for Semi-Automatic Modeling of Complex Environments. In: IEEE 3DIM, pp. 213–220. Ottawa, Ontario, Canada (1997)
13. Karaszewski, M., Sitnik, R., Bunsch, E.: On-line, collision-free positioning of a scanner during fully automated three-dimensional measurement of cultural heritage objects. RAS **60**(9), 1205 – 1219 (2012)
14. Kasper, A., Xue, Z., Dillmann, R.: The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. IJRR **31**(8), 927–934 (2012)
15. Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.K.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE TRA **12**(4), 566–580 (1996)
16. Khalfaoui, S., Seulin, R., Fougerolle, Y., Fofi, D.: View planning approach for automatic 3d digitization of unknown objects. In: ECCV Workshops, *Lecture Notes in Computer Science*, vol. 7585, pp. 496–505. Springer (2012)
17. Kriegel, S., Bodenmüller, T., Suppa, M., Hirzinger, G.: A Surface-Based Next-Best-View Approach for Automated 3D Model Completion of Unknown Objects. In: IEEE ICRA, pp. 4869–4874. Shanghai, China (2011)
18. Kriegel, S., Brucker, M., Marton, Z.C., Bodenmüller, T., Suppa, M.: Combining Object Modeling and Recognition for Active Scene Exploration. In: IEEE/RSJ IROS, pp. 2384–2391. Tokyo, Japan (2013)
19. Kriegel, S., Rink, C., Bodenmüller, T., Narr, A., Suppa, M., Hirzinger, G.: Next-Best-Scan Planning for Autonomous 3D Modeling. In: IEEE/RSJ IROS, pp. 2850–2856. Vilamoura, Portugal (2012)
20. Kuffner, J.J., LaValle, S.M.: RRT-Connect: An Efficient Approach to Single-Query Path Planning. In: IEEE ICRA, pp. 781–787. San Francisco, CA, USA (2000)
21. Larsson, S., Kjellander, J.A.P.: Path planning for laser scanning with an industrial robot. RAS **56**(7), 615–624 (2008)
22. Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., Fulk, D.: The digital michelangelo project: 3D scanning of large statues. In: SIGGRAPH, pp. 131–144 (2000)
23. Liepa, P.: Filling Holes in Meshes. In: ACM SGP, pp. 200–205. Aachen, Germany (2003)
24. Loriot, B., Ralph, S., Gorria, P.: Non-Model Based Method for an Automation of 3D Acquisition and Post-Processing. ELCVIA **7**(3), 67–82 (2008)
25. Low, K.L., Lastra, A.: Efficient Constraint Evaluation Algorithms for Hierarchical Next-Best-View Planning. In: IEEE 3DPVT, pp. 830–837. Chapel Hill, North Carolina, USA (2006)
26. Massios, N.A., Fisher, R.B.: A best next view selection algorithm incorporating a quality criterion. In: BMVC, pp. 780–789. British Machine Vision Association (1998)
27. Maver, J., Bajcsy, R.: Occlusions as a Guide for Planning the Next View. IEEE PAMI **15**, 417–433 (1993)
28. Mehdi-Souzani, C., Thiebaut, F., Lartigue, C.: Scan planning strategy for a general digitized surface. JCISE **6**(4), 331–339 (2006)
29. Pito, R.: A Solution to the Next Best View Problem for Automated Surface Acquisition. IEEE PAMI **21**(10), 1016–1030 (1999)
30. Potthast, C., Sukhatme, G.S.: Next Best View Estimation With Eye In Hand Camera. In: IEEE/RSJ IROS Workshop. San Francisco, CA, USA (2011)
31. Prieto, F., Lepage, R., Boulanger, P., Redarce, T.: A CAD-based 3D data acquisition strategy for inspection. MVA **15**(2), 76–91 (2003)
32. Sahbani, A., El-Khoury, S., Bidaud, P.: An overview of 3d object grasp synthesis algorithms. RAS **60**(3), 326–336 (2012)
33. Scheibe, K., Suppa, M., Hirschmüller, H., Strackenbrock, B., Huang, F., Liu, R., Hirzinger, G.: Multi-scale 3D-Modeling. In: PSIVT, pp. 96–107. Hsinchu, Taiwan (2006)
34. Scott, W.R., Roth, G., Rivest, J.F.: View Planning for Automated 3D Object Reconstruction Inspection. ACM Comput. Surv. **35**(1), 64–96 (2003)
35. Strobl, K.H., Mair, E., Bodenmüller, T., Kielhöfer, S., Sepp, W., Suppa, M., Burschka, D., Hirzinger, G.: The self-referenced DLR 3D-modeler. In: IEEE/RSJ IROS, pp. 21–28. St. Louis, MO, USA (2009)

36. Suppa, M.: Autonomous Robot Work Cell Exploration using Multisensory Eye-in-Hand Systems. Ph.D. thesis, Leibniz Universität Hannover (2008)
37. Suppa, M., Kielhöfer, S., Langwald, J., Hacker, F., Strobl, K.H., Hirzinger, G.: The 3D-Modeller: A Multi-Purpose Vision Platform. In: IEEE ICRA, pp. 781–787. Roma, Italy (2007)
38. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, MA (2005)
39. Torabi, L., Gupta, K.: An autonomous six-DOF eye-in-hand system for in situ 3D object modeling. IJRR **31**(1), 82–100 (2012)
40. Trummer, M., Munkelt, C., Denzler, J.: Online Next-Best-View Planning for Accuracy Optimization Using an Extended E-Criterion. In: IEEE ICPR, pp. 1642–1645. Istanbul, Turkey (2010)
41. Vasquez-Gomez, J.I., Lopez-Damian, E., Sucar, L.E.: View planning for 3D object reconstruction. In: IEEE/RSJ IROS, pp. 4015–4020. St. Louis, MO, USA (2009)
42. Weinmann, M., Schwartz, C., Ruiters, R., Klein, R.: A multi-camera, multi-projector super-resolution framework for structured light. In: IEEE 3DIMPVT, pp. 397–404. Hangzhou, China (2011)
43. Wong, L.M., Dumont, C., Abidi, M.A.: Next Best View System in a 3-D Object Modeling Task. In: IEEE CIRA, pp. 306–311. Monterey, California (1999)
44. Wren, E.: Trend Surface Analysis A Review. CJEG **19**, 39–44 (1973)
45. Wurm, K.M., Hornung, A., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In: IEEE ICRA Workshop. Anchorage, AK, USA (2010)